AD-A120 184 ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB F/G 9/2
AN EXPERT DISTRIBUTED ROBOTICS SYSTEM WITH COMPREHENSION AND LE--ETC(U)
JUL 82 D L WALTZ, R T CHIEN, G DEJONG F49620-82-K-0009
UNCLASSIFIED T-116 AFOSR-TR-82-0879 NL

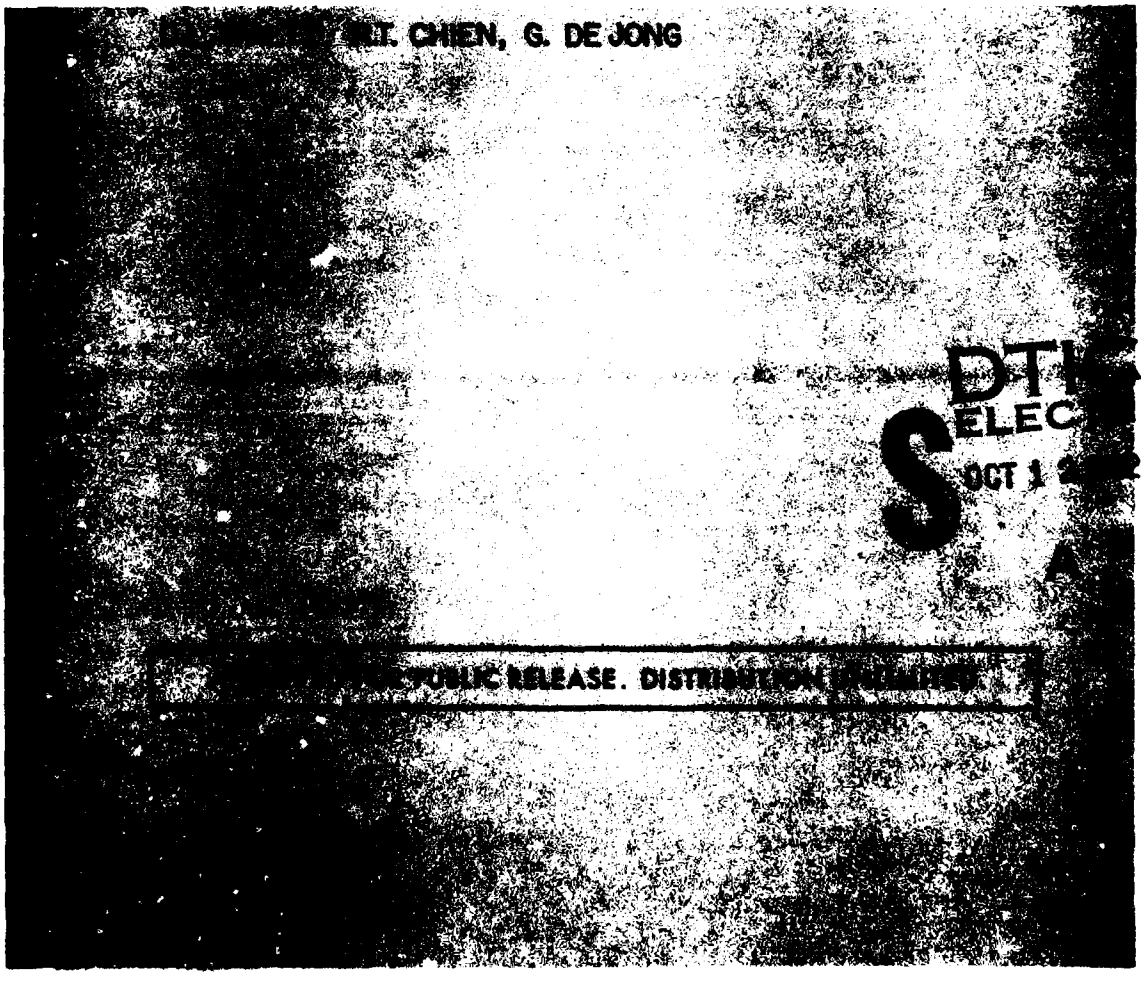
1 OF 1
AD A 120184

END
DATE FILMED
11-82
DTIC

AFOSR-TR- 82-0879
REPORT T-116 , JULY, 1982

COORDINATED SCIENCE LABORATORY

AN EXPERT DISTRIBUTED ROBOTICS SYSTEM WITH COMPREHENSION AND LEARNING ABILITIES IN THE AIRCRAFT FLIGHT DOMAIN



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR-82-0879	2. GOVT ACCESSION NO. AD-A120184	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN EXPERT DISTRIBUTED ROBOTICS SYSTEM WITH COMPREHENSION AND LEARNING ABILITIES IN THE AIRCRAFT FLIGHT DOMAIN		5. TYPE OF REPORT & PERIOD COVERED INTERIM, 1 Jan 82-30 Jun 82 6. PERFORMING ORG. REPORT NUMBER T-116
7. AUTHOR(s) D.L. Waltz, R.T. Chien, G. DeJong		8. CONTRACT OR GRANT NUMBER(s) F49620-82-K-0009
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois 1101 West Springfield Avenue, Urbana IL 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Directorate of Mathematical & Information Sciences Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE July 1982 13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The authors are focusing on in-flight problem diagnosis. Suppose, for example, a pilot simultaneously experiences over-heating in one engine and aileron reverse. He might attribute the problem to the hydraulic system, but unless he possessed detailed technical knowledge of the particular aircraft, he might not be able to decide which sub-assembly component became disfunctional. However, exactly how and where the problem occurred may have implications for how to deal with it. Indeed, a naively plausible, but wrong, assessment of the problem may lead the pilot to exacerbate rather than improve his situation. An (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: on-board intelligent computer system to aid in diagnosis and to suggest corrective measures would be of great help.

There are a number of essential attributes of such a system. First, the system must have a detailed internal model of the particular aircraft. This includes sub-assemblies of the plane, how they relate, and how the plane works as a whole. Second, the system must be able to communicate its diagnoses and suggestions in an efficient and effective manner. It must be efficient because in emergencies time is at a premium. It must be effective or the pilot will routinely reject its advice. This dictates a high level language, perhaps even a natural language, interface between the system and the pilot through which the system can engage in intelligent communication about the problem with the pilot. The system must be able to give its reasons for its assessments and be responsive to counter suggestions from the pilot. Thus, third, the system must have knowledge of its own internal workings. Fourth, the system ought to benefit from its previous experiences and previous discussions with pilots and other experts. Of course, the realization of a system such as this is far in the future. However, the authors believe that significant progress can currently be made towards this goal

Such a system raises a number of theoretically important issues of interest to the investigators. Very broadly, the issues are: (1) developing computer representations for physical mechanisms, (2) intelligent modeling of those mechanisms, (3) high level natural language communication between humans and computers, and (4) learning from experience and instruction.

These issues are very complex and, although they will interact strongly in the ultimate system, initially the must be studied separately. To be tractable, the sub-problems must be of a manageable size. Furthermore, a modern jet aircraft is a phenomenally complicated object. In fact, it is far too involved and complicated to be useable as an initial domain. Thus, the first work is being done on more simplified domains. After gaining experience and understanding of the basic issues involved, the techniques so learned will be applied to the difficult domain of in-flight fault diagnosis.

The remainder of this report is divided into three sections. Each describes the progress reported by one investigator. Several sections contribute to more than one of the four facets of the in-flight diagnosis problem (representation, mechanism modeling, natural language, and learning).

T-116

Semi-Annual Technical Report for Research in

AN EXPERT DISTRIBUTED ROBOTICS SYSTEM WITH COMPREHENSION AND LEARNING ABILITIES IN THE AIRCRAFT FLIGHT DOMAIN

For the Period

January 1, 1982 - June 30, 1982

Submitted to

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Investigators:

D. L. Waltz, R. T. Chien, G. DeJong

July 1982

1. Research Objectives

We are focusing on in-flight problem diagnosis. Suppose, for example, a pilot simultaneously experiences over-heating in one engine and aileron reverse. He might attribute the problem to the hydraulic system, but unless he possessed detailed technical knowledge of the particular aircraft, he might not be able to decide which sub-assembly component became disfunctional. However, exactly how and where the problem occurred may have implications for how to deal with it. Indeed, a naively plausible, but wrong, assessment of the problem may lead the pilot to exacerbate rather than improve his situation. An on-board intelligent computer system to aid in diagnosis and to suggest corrective measures would be of great help.

There are a number of essential attributes of such a system. First, the system must have a detailed internal model of the particular aircraft. This includes sub-assemblies of the plane, how they relate, and how the plane works as a whole. Second, the system must be able to communicate its diagnoses and suggestions in an efficient and effective manner. It must be efficient because in emergencies time is at a premium. It must be effective or the pilot will routinely reject its advice. This dictates a high level language, perhaps even a natural language, interface between the system and the pilot through which the system can engage in intelligent communication about the problem with the pilot. The system must be able to give its reasons for its assessments and be be responsive to counter suggestions from the pilot. Thus, third, the system must have knowledge of its own internal workings. Fourth, ths system ought to benefit from its previous experiences and previous discussions with pilots and other experts. Of course, the realization of a system such as this is far in the future. However, we believe that significant progress can currently be made towards this goal.

Such a system raises a number of theoretically important issues of interest to the investigators. Very broadly, the issues are: 1) developing computer representations for physical mechanisms, 2) intelligent modeling of those mechanisms, 3) high level natural language communication between humans and computers, and 4) learning from experience and instruction.

These issues are very complex and, although they will interact strongly in the ultimate system, initially they must be studied separately. To be tractable, the sub-problems must be of a manageable size. Furthermore, a modern jet aircraft is a phenomenally complicated object. In fact, it is far too involved and complicated to be useable as an initial domain. Thus, our first work is being done on more simplified domains. After gaining experience and understanding of the basic issues involved the techniques so learned will be applied to the difficult domain of in-flight fault diagnosis.

The remainder of this report is divided into three sections. Each describes the progress reported by one investigator. Several sections

contribute to more than one of the four facets of the in-flight diagnosis problem (representation, mechanism modeling, natural language, and learning).

2. Representations and Natural Language Processing

David Waltz
David Spoor
Jordan Pollack
Raman Rajagopalan

2.1. Representing event concepts

It is important for an expert system in the aircraft flight domain to be able to represent events types and specific events in forms that allow a program to (1) recognize when a given type of event is occurring by looking at sensor outputs, (2) express in a comprehensible way what event(s) are occurring at any given time, and (3) understand high-level commands to carry out events sufficiently to fill in all the actions necessary to accomplish the events. We believe that it is important for our systems to be able to use concepts from natural language for these purposes, in order to make such systems maximally comprehensible to their users as well as to their builders and maintainers. We believe that it is also appropriate to use natural language structures as starting points for expressing functional and conceptual structures of mechanisms, given that we want to be able to communicate with our systems about the operation of components at a variety of levels of abstraction. At the same time, we want our representations of components and mechanisms to reflect true causal relationships.

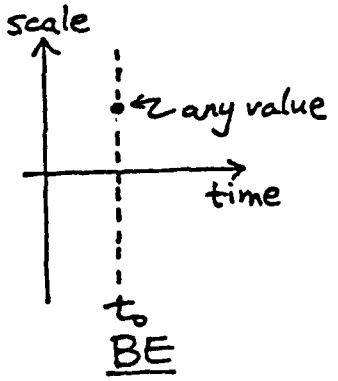
We have therefore been developing representation mechanisms that are capable of encoding information about causal connections, timing, concurrency, rates, durations, forces, quantities,

In their simplest forms, event shape diagrams have a time line, a scale, and values on the scale at one or more points. There are three basic event shape diagrams, illustrated in Figure 1.*

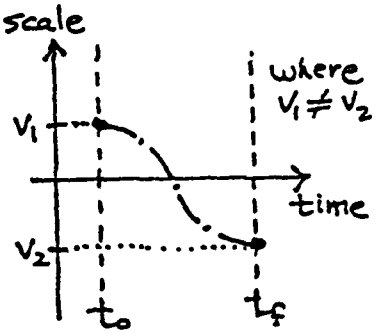
Diagrams can be used to represent concurrent processes, causation, and other temporal relations by aligning two or more diagrams, as illustrated in Figure 2. Figure 2 shows the representation for "eat" Note that four simple diagrams are aligned, and that each has different kinds of scales, and different event shapes. Causal relations also hold between the events described in each simple diagram. The names for the causal relations are adopted from Rieger's CSA work [Rieger 1975]. The action Eating stops in this default case where "desire to eat" goes to zero. "Desire to eat" sums up in one measure coercion, habit, and other factors as well as hunger. Typical values for amounts of food, time required to eat, and so on are also associated with the diagram, to be used as default values.

---

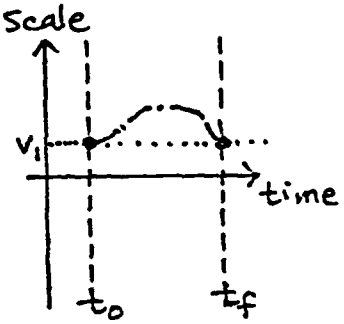
*While diagrams are shown here, data structures to represent these diagrams are very easy to program.



BE

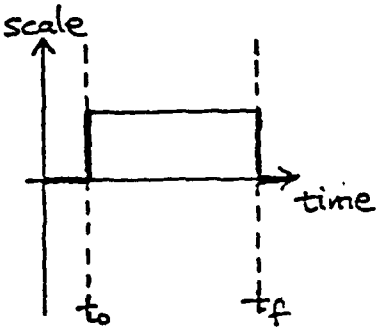


CHANGE



RECUR

Special case:
if scale$(t) = V_1$
for all t, $t_o \leq t \leq t_f$,
then the diagram represents
PERSIST



HAPPEN (if no agent)
DO (if agent)

Figure 1. Basic event shape diagrams.

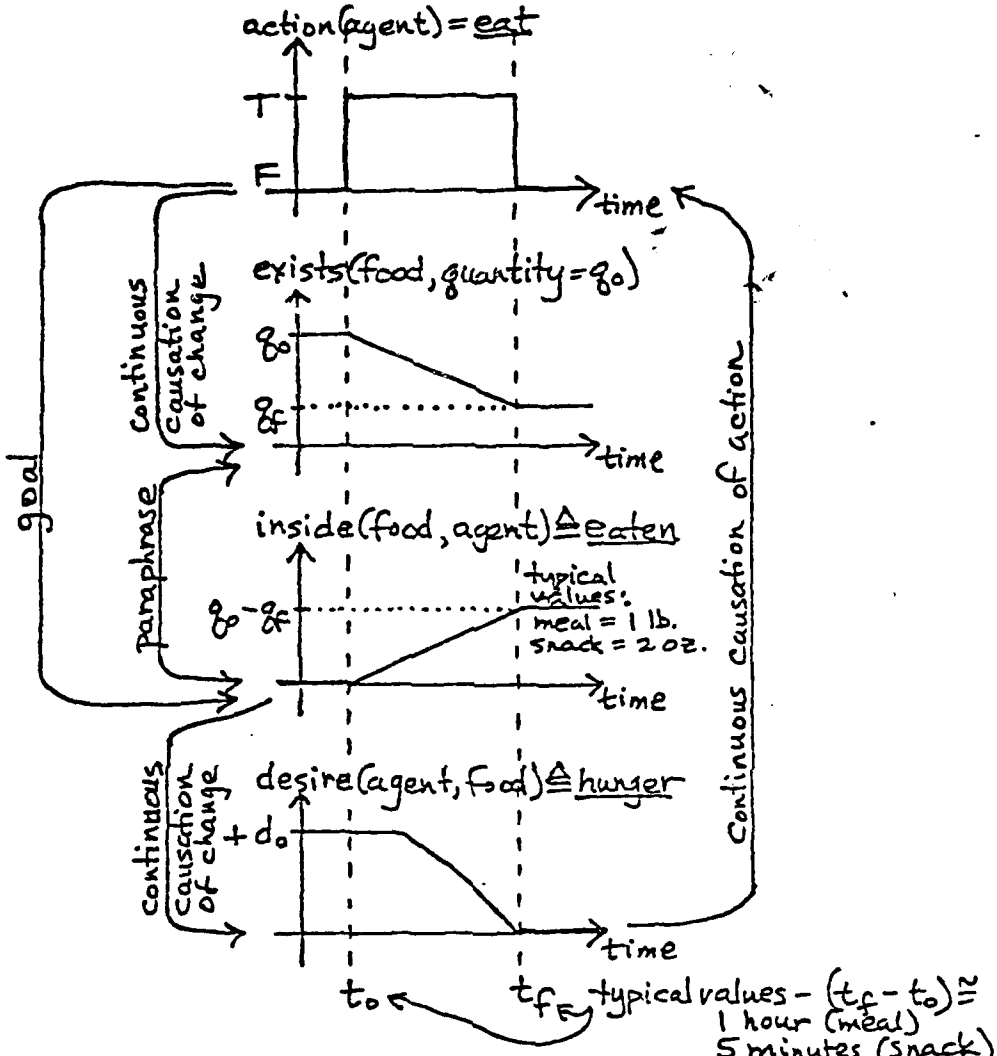


Figure 2. Event shape diagram for eat.

More levels of detail can be added if needed. For instance, the action diagram can be expanded so that eating involves many recurrences of putting food in one's mouth, biting, chewing, and swallowing, and the diagram for the amount of food inside the agent can reflect a series of stepwise changes as each mouthful is ingested.

We are attempting to model various characteristics of physical objects, such as typical uses and typical failures, limits of normal use (i.e. when the device may break down even under ideal operating conditions, expected life, etc.), and causality and connec- tions between separate devices. We have looked at simple models for a battery, a flashlight, and a bicycle pump. Currently we are attempting to model the gas turbine engines typically used in aircraft.

The model for the battery and flashlight takes into account such factors as time and temperature. The battery model included information as to the effects of long term storage, continuous use in a particular device (5 hours of use in a tape recorder as opposed to a flashlight, for example), and the effects of temperature (i.e. a battery which functions well at 70 degrees f. may not function at all at -50 degrees f.). Consideration was also given as to the type of battery (i.e. regular carbon battery, heavy duty battery, or alkaline). The flashlight model used some of the concepts developed by Ken Forbus, such as preconditions of use, causality between various parts, and influences. The model was then further extended to include information concerning timing, and information as to when a particular part may fail. The latter information could be obtained from using models of individual parts, such as the one derived for the battery.

Event Shape Diagrams have been applied to model a bicycle pump. Event Shape Diagrams were found to be good at showing rates of change and recurring events and for giving a sense of elapsed time in an event. They all consisted of continuous lines or curves joined by discontinuities where significant events occurred. Event Shape Diagrams however commit the model to one sequence of events where parallelism of events is possible, as well as fixed levels and rates of change. Thus they model a specific instance of an event well, but do not capture a generic event as well. Causality among events is also weakly modeled. Flow, motion, and other properties dependent upon the physical connection of mechanisms are only indirectly modeled.

2.2. Natural Language Processing

We have been working on a parallel, analogue model for knowledge integration and decision-making in the context of natural language processing. Essentially, the model involves dynamically constructing an unstable weighted network of possibilities, while concurrently sifting and stabilizing the network such that the "best" interpretation is highlighted.

A feasibility study was performed by manually creating a network for the (syntactically) ambiguous sentence:

John ate up the street.

The procedure for setting up the network was based on breadth-first chart parsing [Kay, 1973; Hobbs, 1974]. Using a straightforward mathematical formalism for spreading activation and lateral inhibition [McClelland & Rumelhart, 1980], the network stabilized on the prepositional reading of "up". Next, the weights in the network were slightly adjusted, and the adverbial reading of "up" was selected. Finally, some preliminary semantics were added, at the level of case-frames, and the prepositional reading was selected again. These results are reported in a paper for the 1982 Cognitive Science Conference [Pollack & Waltz, 1982].

The first implementation was used for the feasibility demonstration above, and included mainly syntax knowledge; progress is being made on the second generation program, which will contain both syntactic and case-frame knowledge in a better organized network structure. Furthermore, although the first few generations will work at the sentence level, the goals of this research include the integration of discourse level knowledge.

The parallel organization of processing being developed here is quite amenable to integrated circuit implementation, with its restrictions on physical connectivity [Sutherland & Mead, 1977]. A recent exercise [Pollack, 1982a] demonstrated one possible implementation technique, based on integrating arithmetic with message passing.

While the use of spreading activation and lateral inhibition networks for language processing has been my central concern, this work bears directly on other research within the distributed robotics project. I have built networks which demonstrate repetitive and causal behaviors -- this is directly related to the work on event modeling [Waltz, 1982]. Also, spreading activation has historically been used in memory priming and schema selection techniques [Collins & Quillian, 1972; Ortony, 1976; Fahlman, 1979], so the research on schema acquisition [Dejong, 1982] could be aided by exploration in this area.

2.3. Interlisp

Interlisp-VAX was obtained from USC-ISI in order to provide an alternate environment to Franz Lisp on the CSL VAX. Interlisp-VAX, is equivalent to Interlisp-10, and is now up and available for use here at CSL.

3. Diagnosis and Design of Mechanisms using Deep-Level Understanding Models

R. T. Chien
Bill Frederick
Michael Houghton
Adam Pajerski

3.1. Overall Objective

The purpose of this research is to develop a unified theory of computer-based diagnosis and design of mechanisms through the construction of deep-level under standing models. This approach is based on the premise that neither the table-look-up approach nor the production system approach of the present variety will allow enough in-depth knowledge to be represented, made available conveniently to deal with difficult circumstances often encountered in multiple fault situations. In the aircraft domain redundancy is used so widely that techniques deal with single fault situations only are of little value in practice. Although not our prime objective the theory of design seems to be a natural extension of this approach without too much additional work.

3.2. Why the Other Approaches are Inadequate

The shortcomings of a straight-forward table-look-up approach is quite obvious. Combinatorial explosion prevents us from representing each possible combination in the table especially when many of the variables are analog in nature. Yet sequential procedures are of little help as an intelligent procedure usually requires judgement and therefore logical deduction.

Even though production rules can produce satisfactory results for some applications, their overall effectiveness is limited for several reasons:

(1) Inability to recognize a fault for which a production rules or table entries have been omitted by a human expert designing the systems data base.

(2) Omitted or inconsistent production rules can lead to wrong diagnosis.

(3) Improvements in the power of the system lead to a rapidly expanding set of the necessary production rules.

(4) The system is unable to diagnose multiple faults.

(5) Because of local nature of the production rules algorithm the system cannot explain its failures or propose meaningful tests to clear up ambiguities.

(6) Since the system is a human expert's understanding of the domain it cannot recognize inconsistencies in the provided rules. Inconsistent rules can be introduced if the domain is updated after a lapse of time.

We expect to build a system that deals with all those problems by basing our diagnostic algorithm on the deep-level understanding of the mechanism rather than performing table look-up or precondition matching to arrive at a solution. Through its causal understanding of how subsystems work down to the component level the system's performance should approach that of a human expert. Unlike the production rules approach the system will recognize incompleteness of data necessary to make a diagnosis and will design functional tests within the constraints of the available sensory data. If no such tests are possible the system will explain why there is no solution. The system's extension with changing domain consists only of updating the domain description, limiting the problem of controlling the system's integrity. Because the domain description is the only requirement for system's operation the system is highly portable and can be adapted to a wide range of domains.

3.3. Understanding Deep-Level Semantics

Most of the mechanisms in airplanes are electro-mechanical involving electrical, mechanical and electronic principles. To begin our study with a mechanism of reasonable complexity and which embodies electro-mechanical principles we chose the case study of a refrigerator. It is quite obvious that any theoretical discovery in this investigation would be immediately applicable to most electro-mechanical system either directly or with minor modifications.

3.3.1. Principles of Refrigeration

Refrigeration is the process of moving heat from a space that you want to cool down to a space that you do not care if it gets heated up. The two key words in the definition are heat which refers to some type of heat transfer, and moving which refers to the circulation of a substance that can hold a quantity of heat.

Heat transfer is the natural process of moving heat from a warm substance to a colder substance. There is a rate and direction associated with it that depend on the temperatures and other factors of the substance.

Circulation is the movement of a substance from one space to another, the substance in this case is the refrigerant. The refrigerant moves heat from the space to be cooled to the uncontrolled space. Natural properties of the refrigerant make this possible even though in many cases the space to be cooled is

colder than the uncontrolled space. It also is a physical liquid or gas, so a physical circulation system to contain and move the substance is needed.

The flow graphs in Figure 3 and Figure 4 give a useful representation of the system. These graphs will help to show useful underlying ideas that may not be clear when looking at individual components and their properties. The two graphs show that the important properties that connect all components are heat transfer and refrigerant flow. It is also useful to note that refrigerant flow is proportional to heat transfer by some constant so there is only really one property that connects all components.

The nodes of the graph tell what parts of the system are interacting with each other. The edges tell what type of interrelationship are going on. An equation will be associated with each edge, the equations will be used later when an actual design is performed. Keeping track of the graphs and the fact that the heat transfer and flow at each point are equal will assure a consistent design in the end.

Because heat is not created in this process just moved we will assume that the constant H is about the same at all points. Then taking the second uncontrolled space as a starting point and using the fact that refrigerant can carry heat we get the following flow graph.

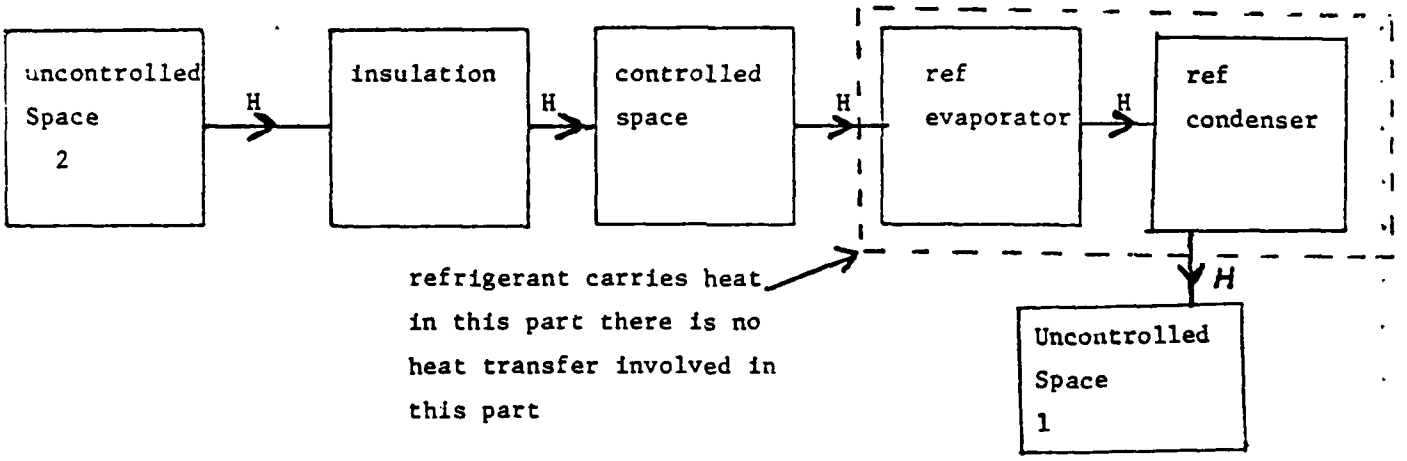


Figure 3

Uncontrolled spaces 1 and 2 represent infinite heat sinks or sources and do not need to be connected.

If you select any point in the circulation system and measure the flow you would find it to be the same at any point. For that reason you can cut it at any point you want and make a flow graph. We will cut it at the compressor because that is the motivation for the flow.

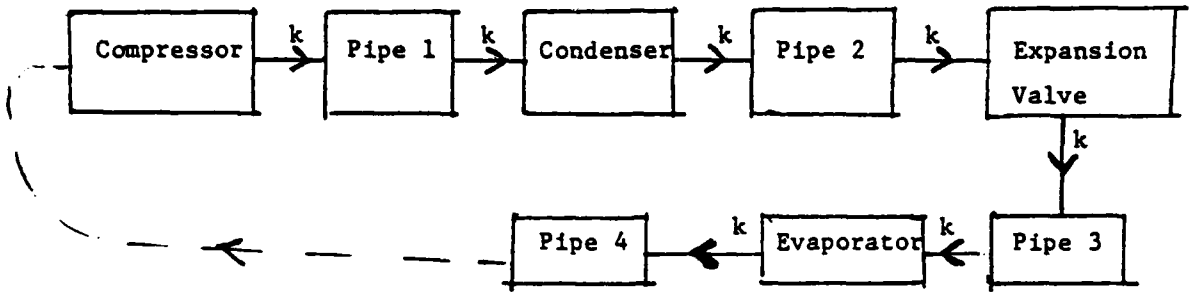


Figure 4

the key for making a graph is

1) break the system up into separate part
2) define important jargon or ideas in each part (example heat transfer, flow, temperature)
3) attach magnitude, direction, etc. to each part for the ideas that apply so a graph can be made

The last set of values defined in the refrigeration circuit, temperature, has no flow associated with it. It can be used as a support value for heat transfer since it is an important indicator of the direction and quantity of heat transferred.

3.3.2. How Design Might Be Done

The key to a useful design is consistency, an example of a consistent flow design is given in the refrigerant flow system where the flow rate is the same at every node. To find underlying principles that will connect all components consistently it is best to list components and their properties and how the components interact with each other with these properties. After that flow graphs as in the last part help to connect the central ideas, the two graphs in the last part show heat transfer and refrigerant flow will connect all components in the refrigeration design. There are equations associated with each edge of the graph and the resulting set of equalities, that result from the fact that

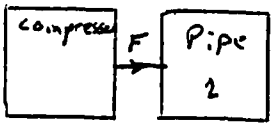
heat transfer or flow at each node is the same, is the first step in design. The next step in the design process is to come up with a specific design by getting values for all unknowns in the equations. This can be done by giving certain unknowns specific values in an attempt to put some constraints on the design, an example of this is setting the outside temperature to some maximum value like 90 degrees. You can also add other equations that do not come from the flow graphs, an example of that is adding cost equations for different types of heat transfer coils with the objective of minimizing cost. Below are the equations that are associated with some of the edges of the graphs and some assumed values for the design. Then we will make a flow chart to carry out the design of a refrigeration system.

flow of refrigerant in system

F = proportional to H

size of pipes and valves is proportional

to F or proportional to H



refrigerant flow through compressor

F = (times rpm displacement)

= proportional to horsepower

rpm = revolutions per min of motor

displacement = volume of refrigerant taken in each revolution

horsepower = motor horsepower compressor

Constraints

surfacearea = constant = S1

insidetemp = constant = TP1

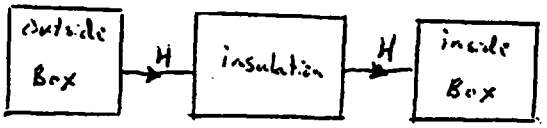
outsidetemp = constant = TP2

(difference insidetemp reftemp1) = constant = TD1

(difference reftemp outsidetemp) = constant = TD2

rpm = constant = RPM

Equations from graphs



heat transfer going through insulation =

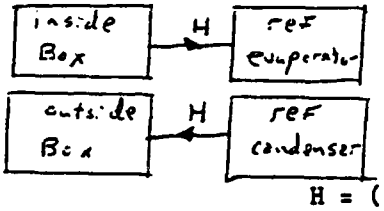
H = (times(times surfacearea (difference outsidetemp insidetemp))heattranscoef)

surfacearea = surface area of box

insidetemp = inside temperature of box

outsidetemp = outside temperature of box

heattranscoef = heat transfer coefficient of the insulation



heat transfer going into or out of a coil

H = (times(times coilarea (difference reftemp outsidetemp))heattrscoil)

H = (times(times coilarea1(difference insidetemp reftemp1))heattrsciol1)

coilarea = area of condenser

coilarea1 = area of evaporator

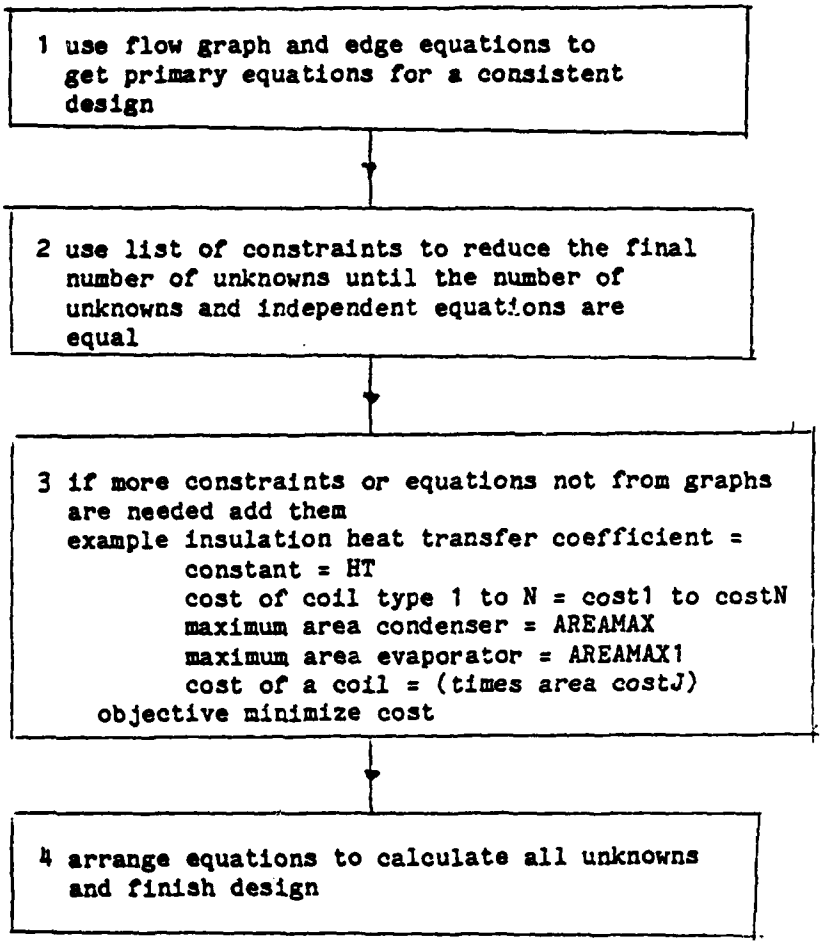
reftemp = refrigerant temp in condenser

reftemp1 = refrigerant temp in evaporator

heattrscoil = heat transfer coefficient cond.

heattrscoil1= heat transfer coefficient evap.

Flow Chart


1 use flow graph and edge equations to get primary equations for a consistent design
2 use list of constraints to reduce the final number of unknowns until the number of unknowns and independent equations are equal
3 if more constraints or equations not from graphs are needed add them
example insulation heat transfer coefficient = constant = HT
cost of coil type 1 to N = cost1 to costN
maximum area condenser = AREAMAX
maximum area evaporator = AREAMAX1
cost of a coil = (times area costJ)
objective minimize cost
4 arrange equations to calculate all unknowns and finish design

3.4. The Question of Representation

The concept of CSAs (Common Sense Algorithm) have been suggested as a universal representation for all mechanism and processes. Although CSAs has been demonstrated to be usable for representing mechanisms like toilets and bicycles, and processes like sawing wood some serious short-comings remain. Aside from the lack of important characteristics as the ability to quantify and include timing information two fundamental weaknesses exist. They are (1) the lack of a hierarchical structure and (2) the lack of a systematic generative procedure.

If several people try to generate a CSA for a given mechanism it is very likely that many different CSAs will result. The mere presence of multiple CSAs for a simple mechanism is not by itself a disturbing fact. What is disturbing is the lack of understanding of the phenomenon. It is a strong indication that the present theory of CSA is by no means complete.

In the technical literature there is a universal accepted way of representing mechanism. One way, we thought, to resolve the question was to devise an automatic procedure to generate CSAs directly from the schematic diagram. We grant you that this approach has the same problem with perspectives. That will have to be dealt with. At least, we will have no question as to what we started with.

Thus the problem is defined as one of devising an algorithm that will accept a technical manual diagram of a device as input and apply engineering knowledge, in a manner similar to that which a paerson would use, to produce a CSA-type computer representation of the device that the computer can use for operation, fault diagnosis, and design of the device. The resulting program will be useful for including new devices in the distributed robotics system.

One of the first questions that must be answered is what kinds of implied engineering knowledge one wishes to include. Specifically, the knowledge required for understanding, for design, and for diagnosis. It is also important to determine what knowledge each of these uses will require.

For understanding, one needs a definition of all components (such as the CSA representation for each component), the specific function of each device in this device, the relationships between different components (the manner in which each component affects the others), and the function the device is intended to perform.

For diagnosis, one needs all the knowledge required for understanding, plus additional information about the failure modes of each of the components, the probability each of the failure modes will occur, and the way in which each of the possible failure modes affects the entire system.

For design, one requires the same knowledge needed for understanding in addition to knowing how changing some components will affect the entire system, the typical ranges of inputs and outputs, and the typical use of each of the components in the system.

Once the definition of all the devices and the implied engineering knowledge is stored, it remains to find an algorithm to apply the implied engineering knowledge to the device definitions to give the desired computer understanding of the device.

4. Knowledge Based Automatic Schema Acquisition

Gerald DeJong
Paul O'Rorke

4.1. Issues involved in schema learning

The concept of knowledge chunks, variously termed schemas, scripts, frames or MOPs has emerged to organize world knowledge in artificial intelligence systems. They have been used to understand natural language, in planning systems, for metaphor processing, memory organization, and story summarizing,

Schemas can be used in an in-flight diagnosis system to represent general solutions to often-occurring problems. Problems that crop up time and again ought to be handled in a more efficient manner than a never-before-encountered problem. For solving novel problems our systems will rely on the general mechanism modeling techniques described elsewhere. Often-occurring problems, on the other hand, ought to be handled without the kind of tortuous reasoning of complete mechanism modeling. Indeed, human experts seem to rely on general "canned" solutions to standard problems. They retreat to a more powerful but *less efficient "reasoning out" process* when they have no appropriate canned solution. We envision a system in which each schema will contain a particular problem solution technique. A schema will be activated when sensors indicate that a problem has occurred for which its particular technique is appropriate. Each will contain the patterns of values indicating a given problem, as well as the steps necessary to solve the problem. Thus, many different but similar problems will map onto the same schema.

There has been little work on how schema constructs are acquired; in most AI systems programmers simply "build in" the requisite knowledge structures. This is an arduous, time-consuming process and is especially inappropriate in a domain such as in-flight diagnosis. Pilots can be trained to handle predictable failures of the kind listed in fligt manuals. To be useful, an on-board AI system must be able to handle multiple simultaneous failures and unpredictable subtle failures as well. If one of these novel failure patterns recurs, the system should be ready for it on succeeding occasions. That is, the system ought to construct and save new schemas it develops. In so doing it might even notice systematic troubles and similarities that would escape humans.

In our study of schema learning thus far we have identified four types of event situations in which a system ought to construct a new schema. They are:

1) Schema Composition
2) Secondary Effect Elevation
3) Schema Alteration
4) Volitionalization

Schema composition involves the synthesis of a new schema from two or more pre-existing schemas. Typically, the preconditions of one schema are satisfied in a non-trivial way by the other. In secondary effect elevation an incidental post-condition (or side effect) of a schema is elevated to the status of a main effect and the remainder of the schema is altered to eliminate now superfluous steps. Schema alteration consists of creating a news schema by adding a constraint in the execution of an existing schema and adjusting the rest of the schema to be consistent with it. Volitionalization yields a new schema by breaking a non-volitional causal connection somewhere in the existing schema and adding the potential, at that point, for human intervention.

4.2. An Implementation

We have been working on an implementation to explore, first, volitionalization.

Our initial learning studies have been conducted in the natural language processing domain, with a schema-based story processor as the performance element. The stories we have considered are sequences of sentences which describe simple events. Schemas are packages of knowledge usually associated with such events. Schemas also represent states-of-affairs (like possession) and themes (like avarice). Stories are processed by applying known schemas to the events observed in the stories. The result of processing a particular story is a network of nodes and links. Nodes represent schemas of different levels of abstraction, from "Agrippina possesses the estate" to "Agrippina inherits the estate from Claudius." Links represent relations between the nodes (eg. precedes, leads-to, part-of).

We believe it is important to realize examples of explanatory schema acquisition, so we have analyzed an event which occurs often in mystery stories. People kill benefactors or heirs apparent, in order to inherit bequests. This corresponds to a volitional schema which might be called "facilitation of inheritance by murder". Explanatory schema acquisition enables a system to learn this plan for obtaining possession from more basic schemas about inheritance and murder.

A simple story which captures the essential details of the example is:

1. Claudius owned an island estate.
2. Agrippina fed Claudius some poisoned mushrooms.
3. Claudius died.
4. Agrippina inherited the island estate.

Before processing this story, the system does not possess the "facilitation of inheritance by murder" schema. However, it does have schemas for murdering, inheritance, and eating. These schemas include all of the system's knowledge about these concepts. For example, it knows about ownership and dying and how these relate to inheritance. Furthermore, the

system knows about goals people can have and how they influence their actions and how they relate to other goals.

In the course of processing this story, the system acquires the facilitation of inheritance schema. Input (1) is processed as background material. It does not report an action, merely a state of affairs that is true at the beginning of the story. Input (2) is recognized as an intentional murder of Claudius by Agrippina. Yet the system realizes that this input is not fully processed. Intentional actions, such as murdering someone, require motivations. Since the system knows of no motive for the murder, it is marked as requiring further explanation. Input (3) is processed as an anticipated effect of the murder. The system can predict that Claudius will probably die from its schema for murder. Input (4) is understood as an instance of an inheritance. This is processed with the system's inheritance schema. This input requires further processing as well. The system realizes that the inheritance is a gain for Agrippina and has the necessary precondition of Claudius dying. Since Agrippina was responsible for Claudius's death, the system assumes that Agrippina knew she would benefit via the inheritance and that the poisoning was motivated by her desire to own Claudius's estate. Thus, the system now has a complete explanation for the actions in the story. The understanding phase is finished.

Now the internal representation for the explanation of the story is generalized into a schema. The system suspects that a new schema can be constructed because a) the no single existing schema could process the story and b) the novel combination of known schemas fits one of the explanatory schema acquisition patterns (that of "volitionalization"). The generalization procedure consists of relaxing constraints on the objects, actors, and actions in the explanation while preserving the underlying structure of the explanation. For example, the system realizes that the people involved need not be Agrippina and Claudius. Rather any two distinct individuals will do so long as the first is the beneficiary of the second. Using this kind of reasoning the system constructs the facilitation of inheritance schema from observing a single instance of it.

In more detail, the story processor "understands" this story by applying knowledge given to it in the form of schemas like the following.

```
(naive-individual-inherit
    (vars (benefactor       ?person1
           prior-heirs      ?people
           bequests         ?objects
           heir             ?person2))
    (complex (nii1 (poss (actor ?person1) (object ?objects))
              nii2 (death (actor ?person1))
              nii3 (nii-conditional-transfer))
             precedes ((nii1 nii2))
             leads-to ((nii2 nii3))))
```

For example, the fourth statement in the Claudius story is processed by

instantiating the given naive-inheritance schema. A node is created which stands for the instance of naive-individual-inherit obtained by identifying Agrippina as the heir and the island estate as a bequest. Inheritance is a complex event. The "complex" slot of the inheritance schema tells the story processor to activate three subnodes and establish links representing part-of, precedes, and leads-to relations. Instantiation is recursive, so that a network of new nodes is established for each new input, with connections to the old network through recognition of old events. When the story processor attempts to activate nil1 (poss (actor ?person1) (object (island estate)), it checks the active nodes in the network already constructed from the previous three sentences. Nil1 is matched with the node created on processing the first input to yield the inference that Claudius is the benefactor in the inheritance. The death of Claudius is also recognized as an old event, not only because it was explicitly mentioned in the story, but also because the poisoning of Claudius by Agrippina activated the more general murder schema, and both called for Claudius' death. The third subevent is also complex, and expands to a sub-network which captures the notion that if all the prior-heirs are dead, then Agrippina benefits from a transfer-of-possession of the island estate. The node representing Agrippina's possession of the estate hypothesizes a link to a node representing the theme of Greed/Materialism.

Notice that the inherit schema was *passive* or *non-volitional*. It said nothing about goals, and specified no planner. Other schemas like murder might well be called schemes. Murder is a volitional schema which may be viewed as a plan whereby the murderer achieves the immediate goal of the death of the victim.

Certain paths through the network of nodes constructed by the story processor can be viewed as explanations. The most important explanation chains are intentional explanations, they explain actions in terms of themes. The construction of these chains is the heart of the story processor's "understanding" process. For example, Agrippina's act of feeding Claudius poisoned mushrooms is ultimately explained by the story processor as a case of poison/murder which caused Claudius' death which led (through the inheritance schema) to possession of the island estate by Agrippina. The explanation is completed by a link from this possession node to a node representing the theme of Greed/Materialism, where the buck stops.

The role of explanatory schema acquisition is to enhance the performance of the story processor by constructing new schemas. If the story processor had a special schema for "facilitation of inheritance by murder", the processing of the Claudius story might be trivial. Certainly, the explanation for Agrippina's action would be much simpler. Instead of a rather long and complicated explanation chain through a network of schemas including inherit and murder, the explanation for Agrippina's action would be that it was part of a plan for achieving the goal of possessing the estate, which she wanted because she was Greedy or Materialistic. Explanatory schema acquisition can improve story processing by providing such schemas.

The mystery story example of "benefactor elimination" has spurred us to significant progress. Progress has been made on a representation language for schemata and on a schema-based story processor. More important from the standpoint of machine learning is the fact that this example led us to an important general explanatory schema acquisition method which we call volitionalization. Volitionalization is the process which makes a state in a passive schema into the goal of a new volitional schema. The new volitional schema is constructed from the passive schema and from whatever volitional schemas are required to achieve the goal. Cases where volitionalization is applicable are easily recognized. The story processor can invoke it whenever an intentional explanation of an action leads to a state desired by the actor in an ordinarily passive schema.

Volitionalization exemplifies the potential relevance of this work on explanatory schema acquisition to expert distributed robotics systems as proposed in []. Volitionalization makes it possible for such systems to understand otherwise senseless action as "part of a plan". A problem solver such as an in-flight diagnostician capable of volitionalization could learn new plans by observing and understanding actions of human teachers.

For example, suppose an in-flight diagnostics system for the space shuttle observed the sequence of events:

1. Space shuttle x was conducting a test in outer space.
2. The cargo bay was extremely cold because no sunlight shone on it.
3. The cargo bay door of space shuttle x failed to close.
4. Mission control ordered the pilots to re-orient the shuttle so that sunlight shone on the cargo bay.
5. The temperature of the cargo bay rose.
6. The door of the cargo bay became unstuck and successfully closed.

With the proper schemas, with information about temperatures in space and the reaction of objects to extreme temperatures, a schema-based diagnostician could explain the observations:

A. The cargo bay door failed to close because it was warped by the extreme cold of outer space.
B. The cargo bay door became unstuck because it straightened out after being warmed by the sun. It was warmed by the sun because the pilots re-oriented the shuttle because they were ordered to do so by Mission Control. The actions of both Mission Control and the pilots could ultimately be explained by the theme "people like their devices to function properly". If more basic themes are desired, one could continue the intentional explanations. Closing the shuttle bay door may well be a necessary condition for the safe return of the astronauts.

A system capable of explanatory schema acquisition could apply volitionalization to this example because actions were taken to make an

ordinarily non-volitional event happen. The usually passive warming of the cargo bay door was turned into a goal because it might restore the door's functionality. The actions taken to volitionalize the passive "warming-by-sun-in-space" schema included "re-orient-shuttle" and "transmit-order". The result of applying volitionalization to such examples is a plan for shooting troubles caused by reversible changes in materials due to temperature. A trouble shooter capable of volitionalization might well be able to apply the schema learned from the example above to understand novel situations which its programmers had not anticipated.

5. References

Collins, A. and M.R. Quillian, 'Experiments on semantic memory and language comprehension' in L.W. Gregg (Ed.) Cognition in Learning and Memory, Wiley, New York, 1972.

DeJong, G., "Prediction and Substantiation: A new approach for Natural Language Processing" Cognitive Science V.3 #.3 pp. 251-273, 1980.

Dejong, G., "Explanatory Schema Acquisition",To Appear in Proc. NCAI, Pittsburgh, August, 1982.

De Kleer, J., "Qualitative and Quantitative Knowledge in Classical Mechanics", AI-TR-352, AI Lab, MIT, December 1975

Fahlman, S.E., NETL: A system for Representing and Using Real-World Knowledge, MIT Press, 1979.

Forbus, K., "Qualitative Process Theory", AI Memo #664 MIT AI Lab, February 1982

General Electric Aircraft Gas Turbine Guide, Aircraft Engine Group, AEG-607R (10/80)

Hobbs, J.R., "A Metalanguage for Expressing Grammatical Restrictions in Nodal Spans Parsing of Natural Language.", Report NSO-2,Courant Institute, NYU, January 1974

Kay, M., "The MIND System", in Rustin (Ed.) Natural Language Processing, Algorithmics Press, New York, 1973.

McClelland, J.L. and D.E. Rumelhart, "An Interactive Activation Model of the Effect of Context in Perception", Technical reports 91 & 95, Center for Human Information Processing, UCSD, 1980.

Ortony, A., "SAPIENS: Spreading Activation Processing of Information Enclosed in Associative Network Structures", unpublished, 1976.

Pollack, J., "An Activation/Inhibition Network VLSI Cell", WP #31, Advanced Automation Group, Coordinated Science Laboratory, Urbana, January, 1982a

Pollack, J. & D. Waltz, "Natural Language Processing Using Spreading Activation and Lateral Inhibition," To Appear in Proc. Cognitive Science Conference, Ann Arbor, August, 1982.

Sutherland, I.E. & C. A. Mead, "Microelectronics and Computer Science," Scientific American, V. 237, N. 9, September, 1977, pp. 210-228.

Waltz, D.L., "Event Shape Diagrams", To Appear in Proc. NCAI, Pittsburgh, August, 1982.

6. Publications A paper titled "Understanding Novel Language" by Gerald DeJong and David Waltz describing some of this research will appear in The International Journal of Computers and Mathematics.

7. Personnel

There are three investigators on the project: Professor David Waltz (principle), Professor R. T. Chien, and Professor Gerald DeJong. Besides the investigators, there are seven graduate students contributing to the project: Jordan Pollack, Dave Spoor, Raman Rajagopalan (advisees of Professor Waltz), Paul O'Rorke (advisee of Professor DeJong), Adam Pajerski, William Frederick, and Michael Houghton (advisees of Professor Chien).

8. Interactions

Since the project has been underway for just six months, there are as yet no interactions to report.

9. Inventions and patent disclosures

There have been no inventions or patents stemming from this research.

10. Other Statements

Please find attached a copy of the journal article by DeJong and Waltz mentioned under "publications".

Understanding Novel Language*

Gerald F. DeJong and David L. Waltz

Coordinated Science Laboratory and
Electrical Engineering Department
University of Illinois
Urbana, IL 61801

1. Introduction

Natural language understanding systems are interesting to the extent that they understand material that they were never explicitly programmed to handle. A system such as ELIZA (Weizenbaum (1966)) or PARRY (Colby et al (1974)), which operates primarily by pattern matching, is less interesting than a system which has a set of general rules that can be used to generate a meaning representation for unanticipated inputs. There are a wide variety of types of unanticipated input. Some examples are:

a. New instances of known case frames, scripts, or plans. Each of these can be a kind of novel language in the sense that sentences never seen before can be processed appropriately. This may mean that information is retrieved from a data base on request, or that a representation of a news story is constructed and remembered, or that a question is answered about an earlier dialogue, and so on. If the general rules in a system are good ones, then a relatively small number of rules will allow a program to handle a wide

*This work was supported in part by the Office of Naval Research under contract N00014-75-C-0612, in part by the Air Force Office of Scientific Research under grant F49620-82-K-0009, and in part by the National Science Foundation under grants NSF IST 81-17238 and NSF IST 81-20254.

variety of inputs, most of which were never explicitly anticipated by the programmer of the system. This is the simplest type of novel language, and is by now so familiar that it hardly seems to be a way of dealing with novel language at all.

b. Isolated novel words that have to be understood in context. Some work has been done in this area by Granger (1977). Whenever we can extract a meaning structure for a sentence in context, we have some hope of guessing the meaning of a novel word. For example, if we were told:

When the tank got low, John filled his car with gasohol.

A system that had some scriptal knowledge in the automobile domain could guess that gasohol was a kind of fuel, or possibly a fluid to substitute for oil, water, or antifreeze, or by some stretch of the imagination, gasohol might be something to put in a tank that just happens to be being transported by the car. Several types of information can be used to constrain the possible meanings for gasohol: it is something that can be the instrument of "fill", something that a car is filled with, probably its tank, that since the tank got low, something, probably the car or John, was using up the substance in the tank.

c. Combinations of words that denote items never before known to a system. Examples in a) above shade into others where concepts are referenced that are novel to a system. For example, complex noun phrases can use familiar words to construct novel items, as in the phrase (from Finin (1980)):

...engine housing acid damage report summary...

Here, all the words (engine, housing, etc.) may be known, but the phrase

taken as a whole denotes an item that may never have been encountered before by the system. A program that "understands" this phrase could create an internal representation for the item, and infer properties about the item, e.g. that the item was the summary part of a report, that the report was about engine housing acid damage, that the material of the engine housing is probably metal, that the acid damage was to the housing, that acid damage to metal is called "corrosion", and so on. From this information, a system could recognize paraphrases and a variety of references to the same item.

d. Events that are novel, as in the example:

My dachshund bit our postman on the ear.

Waltz (1981) lays out mechanisms that would allow a system to generate the working equivalent of a mental image for this sentence, attempt to simulate the running of a "mental image" corresponding to the sentence, and from the difficulties encountered in running the mental image simulation, judge that the sentence was at least mildly implausible.

e. New schemas, describing goal-oriented sequences of actions that may never have been encountered before, as in hearing and understanding the nature of skyjacking for the first time (DeJong (1982). Here, the understanding consists of first untangling the motivations for each of the participants, accounting for each of the actions that are part of the overall schema, and generalizing the schema so that novel occurrences of similar schemas can remind the system of the original schema.

f. Novel metaphors and analogies. Here the variety of language that

requires explanation is staggering. Understanding metaphorical language first requires noting that the language is metaphorical, that is that it couldn't be literal descriptive text. (This in turn requires an internal model of what is ordinary, expected, or possible, that a system can use to judge the plausibility of novel language -- see item d) above.) Next, information from the "base domain", that is the domain in which the language has literal meaning, must be somehow transferred (with appropriate modifications) to the "target domain", that is, the domain which is actually being described. As an example, given the sentence:

John ate up the compliments.

we would want to transfer material such as pleasure, desire, and "ingestion" (suitably modified) from the eating domain to the communication domain. The result can become the basis for learning about a new abstract domain or it may simply be that a metaphor allows one to express in a few words many notions about a target domain that would otherwise require a much lengthier exposition. In any case, a system should also keep some record of its metaphor understanding process, so that subsequent processing of similar metaphors would be eased.

In this article, we look in more detail at the problem of designing mechanisms that will allow us to deal with the types of novel language described in e) and f) above, namely schema learning and the understanding of metaphors. This work is just beginning. The examples we describe have been chosen to be types that occur commonly, so that rules that we need to understand them can be used to also understand a much wider range of novel language. However, we must note that there is only so far that rules can

take us: ultimately the power of systems will depend on the sheer amount of knowledge they have, knowledge which can be used as the base domain for new metaphors, and schemas that can be used to build yet more schemas. Therefore, to really achieve something resembling common sense, we will have to exercise our rules on whatever base information we have, building a yet larger base on which the rules can operate recursively. This important process is meant to be a first-order model of the process of adult knowledge acquisition through language.

2. Schema Learning

In this section we examine the problem of processing texts that express unfamiliar concepts. Acquiring some grasp of those new concepts is an essential aspect of processing such texts. This is different from learning new words from context. The distinction here is between unfamiliar words that express familiar concepts and familiar words that express unfamiliar concepts. The former problem has been somewhat studied (Selfridge, Granger, Anderson, Langley). The latter has not.

How can familiar words express unfamiliar concepts? After all, knowing a word entails knowing the set of concepts corresponding to its various word senses. While this is true, words in aggregate often can be used to express concepts beyond the simple composition of their meanings. These larger concepts have variously been termed frames (Minsky (1974), Charniak (1976)) or schemas (Bobrow and Norman (1975), Chafe (1975)) or scripts (Schank and Abelson (1977)) or MOPs (Schank (1980)). Structures

corresponding to these larger concepts are used to organize world knowledge in artificial intelligence systems, and play a crucial role in the understanding process in natural languge systems (for example, see Cullingford (1978), Charniak (1977), Bobrow et al. (1977), Wilensky (1978), DeJong (1979)). We will use the (relatively) neutral term "schema" to refer to these knowledge structures.

Very briefly, schemas are used in natural language processing as follows. A text is input to the system. The schemas relevant to the situations described in the text are selected and activated. Schema selection is a difficult problem, outside the domain of this paper. There have been several approaches (e.g., Charniak (1978), DeJong (1979), Fahlman (1979).

After schema activation, text sentences are interpreted with respect to the chosen schemas. For each situation the corresponding schema supplies normal causal and temporal connections among events, a specification of what is important and what is not, preconditions and postconditions, etc. Thus, the use of schemas facilitates the task of constructing a unified conceptual representation for the text as a whole. In some systems (DeJong (1979), Lebowitz (1980)) the schemas are also used to aid in word and sentence interpretation.

Now we can ask a crucial question: What can a natural language system do if it does not have an appropriate schema for understanding a new input text? As a partial answer, we will introduce a new kind of learning called Explanatory Schema Acquisition. As the name implies, it is used to acquire schemas. It is not a universal learning technique. The method will be

applied only to acquisition of volitional schemas, i.e., schemas used by people in problem solving situations. Furthermore, it builds on knowledge already in the system and so it is not immediately applicable to learning a system's first schemas. Even with non-schema and first schema learning ruled out, a very large and interesting class of learning remains. In fact, it seems that a very large fraction of human adult learning is of this kind. It encompasses learning schemas from instruction, from observation of others, from untutored examples, and from fortuitous accidents.

The main argument that will be advanced is that acquiring schemas involves generalizing structures made up of old and familiar schemas which are combined in novel ways. The generalizing process itself is performed through consideration of the interactions between the effects, preconditions and slot filler constraints supplied by the component schemas.

Thus, the method is a knowledge based one. It is capable of one trial learning. Moreover, it relies very little on inductively acquired correlational experience.

2.1. An Example

To clarify the procedure, consider an example. This example is a story about a kidnapping. Let us assume that we, the readers of this example, do not yet have a schema for kidnapping or extortion or any similar notion. We do, however, assume the knowledge of a considerable quantity of background information about stealing, bargaining, the use of normal physical objects, and goals of people and institutions.

Example story:

> Paris police disclosed Tuesday that a man who identified himself as Jean Maraneaux abducted the 12 year old daughter of wealthy Parisian businessman Michel Boullard late last week. Boullard received a a letter containing a snapshot of the kidnapped girl. The next day he received a telegram demanding that 1 million francs be left in a lobby waste basket of the crowded Pompidou Center in exchange for the girl. Asking that the police not intervene, Boullard arranged for the delivery of the money. His daughter was found wandering blindfolded with her hands bound near his downtown office on Monday.

A KIDNAPPING schema, if we had one, would contain information to help us make sense of the story. With it, processing the story would be relatively easy.

But by assumption we do not know about kidnapping. Therefore some events in the story are incomprehensible. In particular we cannot explain why Maraneaux might steal Boullard's daughter. While this is quite clearly an instance of taking something that belongs to someone else, there is no motivation for it. The daughter has no apparent value to Boullard; a person, unlike money, cannot be used to acquire other valued goods. Any schema-based understander requires motivations for major volitional actions (such as a character invoking the STEAL schema). Therefore, this input seem anomalous.

The confusion is resolved by the next sentence. This input invokes the BARGAIN schema. We know immediately the motivation for Maraneaux trying to bargain with Boullard: he is trying to acquire money. Possessing money is a common goal that can be attributed to most people. Thus, it serves as an understandable motivation for the bargaining. Furthermore,

For example, the fourth statement in the Claudius story is processed by

stealing the girl is now motivated: Maraneaux used the STEAL schema to satisfy a precondition of the BARGAIN schema. The precondition states that the bargain is unlikely to work unless each party indeed possesses the item he plans to trade away.

Thus far we have done nothing new. Previous systems have proposed understanding new text inputs via analysis of goals and plans of the characters (Wilensky (1978), Charniak (1976), Schmidt and Sridharan (1977)) These systems tend to be more oriented toward "planning" or "problem solving" than "script application."

Once the story has been understood in this way it might already be viewed as a new schema. The system could file away the representation as a method by which a particular person (Maraneaux) can procure a particular amount of money (one million francs) by a particular action (stealing Boullard's daughter and offering to trade her back for the money). This is a mistake for several reasons. The most important is that it is simply far too specific.

Our concern here is how a system might do better than to simply file away a very specific plan. Our contention is that the same knowledge used to process the input in the first place can be used to make the schema more general. For example, the system has the knowledge necessary to prove that if Maraneaux wanted one hundred thousand francs instead of a million, that the same plan would work. It can do this because the system knows the function of the million francs in Maraneaux's plan. It knows that the money is traded by Boullard for the return of his daughter. Also it knows

that the preconditions for Boullard's acceptance of the proposed bargain are that 1) Boullard must value his daughter's safety more than the money and 2) that Boullard must have access to that amount of money. Clearly, since one million francs satisfies these requirements, any amount less that one million francs also satisfies the requirements and would have worked. Sums larger than a million francs might work as well provided they do not violate (1) or (2) above. We have been a bit sloppy in our analysis. To understand Maraneaux's actions it is not important in reality for Boullard to have access to the money but only for Maraneaux to <u>believe</u> he does, and for Maraneaux to <u>believe</u> Boullard values his daughter. Nonetheless, the point is well made: this event can be generalized through knowledge-based manipulations using information that had to be in the system anyway in order for the story to be understood. In a like manner the identity of Boullard, his daughter, and Maraneaux are not important. What is important are that these roles be played by people with certain relationships to other people and things. The required relationships are dictated by the volitional actions required of the people by the schema. After these knowledge-based generalizations have been made, the specific event can be transformed into a *KIDNAP* schema.

In general, the newly generalized schemas require further refinement. Due to eccentricities in the input story, the schema may lack information. For example, if the first kidnapping story seen by the system reported the kidnappers successfully escaping with the ransom even though they killed the hostage, the system might acquire a distorted concept of kidnapping. Even more frequent are cases where the first schema constructed is correct but incomplete. This might result from situations where there are

alternate methods of achieving certain sub-goals, only one of which is reported. Clearly, schema modification is essential. Thus, the system's schemas must constantly be adjusted and refined in reaction to normal input processing.

2.2. The Generalization Process

There are two problems that the generalization process must face. The first is to know when it should be applied. Clearly, every input text ought not to cause the system to construct a new schema. Only "interesting" inputs should invoke the schema acquisition system. The second problem is how to perform the generalization. There are a number of subproblems here, for example, selecting which events and objects should be generalized, imposing limits on the extent of generalization, and actually carrying out the schema modification.

There are four situations which when recognized in the text either individually or in combination ought to invoke the generalization routines. They are:

Schema Composition
Secondary Effect Elevation
Schema Alteration
Volitionalization

In the first part of this section we will illustrate each of these situations with an example.

2.2.1. Schema Composition

The first situation we will discuss is called schema composition. Basically, it involves composing known schemas in a novel way. Typically, this will involve a primary schema, essentially unchanged, with one or more of its preconditions satisfied in a novel way by other known schemas.

An example of this was seen in the above kidnapping story. In that story, the primary schema is BARGAIN, a schema which we assumed the system already knew. One of the preconditions specified in the BARGAIN schema is that each party to the bargain must convince the other that he can indeed deliver his side of the bargain. For Maraneaux, this corresponds to making Boullard believe that he (Maraneaux) has control of Boullard's daughter and can, therefore, relinquish the girl to him. Maraneaux achieves this by actually establishing control over the daughter (via an instance of the STEAL schema) and then sending Boullard a photograph. To the system, this is a novel way to satisfy BARGAIN's preconditions. We know this must be novel to the system because if it were not, the system would already have a schema in which this precondition of BARGAIN was satisfied by an application of STEAL. But by hypothesis, the system does not yet possess a kidnapping schema and therefore, cannot yet know of this method of satisfying the precondition. Thus, a precondition of a known schema has been satisfied in an interesting new way, and a new schema must be constructed to capture the underlying generalization.

2.2.2. Secondary Effect Elevation

Consider the following scenario:

> Fred wanted to date only Sue, but Sue steadfastly refused his overtures. Fred was on the verge of giving up when he saw what happened to his friend, John: John wanted to date Mary but she also refused. John started seeing Wilma. Mary became jealous and the next time he asked her, Mary eagerly accepted. Fred told Sue that he was going to make a date with Lisa.

Here Fred has not acquired a new schema; he has used an existing schema (DATE) in a new way. This is called secondary effect elevation. Fred's DATE schema already contains all of the knowledge necessary for resolving his dilemma. The problem is that the normal DATE schema is organized in the wrong way. In secondary effect elevation situations an existing schema is annotated indicating that the schema may be used to achieve a result which is normally neutral or negative.

The main purpose of the DATE schema is to satisfy certain recurring social goals (like companionship, sex, etc.). DATE contains secondary effects as well. These are often undesirable effects accompanying the main, planned effects. For example, one is usually monetarily poorer after a date. Another secondary effect is that if one has an old girlfriend, she may become jealous of a new date.

What Fred learned from John's experience is that it is occasionally useful to invoke the DATE schema in order to cause one of its secondary effects (jealousy) while completely ignoring the usual main goal.

Just as with schema composition, the existing schema is changed to reflect a generalization made from a specific instance. In this case, the specific instance is John's interactions with Mary. Notice, however, that Fred did not simply copy John's actions. John actually made a date with Wilma while Fred only expressed an intention to date Lisa. This is not an earth-shaking difference, but in the context of dating it is extremely significant. In the normal DATE situation expressing an intention to date someone is not nearly so satisfying as an actual date. Once modified for the purpose of causing jealousy, however, expressing an intention for a date and actually carrying it out can be equally effective.

One might argue that the distinction between main and secondary effects of a schema is otiose and, in situations such as this, even deleterious. After all, DATE already had all of the information necessary for solving Fred's problem. If a system simply treats all of the effects of a schema the same, then any effect can be singled out during the planning process to be used as the main goal. There is, however, a strong argument against this position. The possible desired effects of a schema do not exist only within the schema itself. They are used to organize and select among schemas in both understanding and planning applications (see Charniak Ms MAL and frame selection). Many effects (like feeling more tired after a date than before) will not be used in the normal planning or understanding process. If they are treated the same as legitimate main goals the system will be swamped in a combinatorial quagmire of undifferentiated possibilities, most of which are wildly implausible. For example, we do not want our understanding process to predict that John will take a nap when it it is told that John dated Mary. Given the input "John took a

nap" the system ought to be able to justify it. However, it ought not actively predict it. Given the multiplicity of individual actions making up the DATE schema (each with its own set of effects) the vast majority of the effects from this schema (and any other schema) are simply irrelevant to overall planning and understanding processes. Instead, we would like our system to single out the plausible volitional effects of its schemas and use only those for schema organization and selection. Thus, in our example, Fred has constructed, via secondary effect elevation, a new use of the DATE schema.

2.2.3. Schema Alteration

Schema alteration involves modifying a nearly correct schema so that it fits the requirements of a new situation. The alteration process is guided by the system's world model. This is illustrated by the following brief anecdote:

> Recently I had occasion to replace temporarily a broken window in my back door with a plywood panel. The plywood sheet from which the panel was to be cut had a "good" side and a "bad" side (as does most raw lumber). The good side was reasonably smooth while the bad side had several ruts and knot holes. I automatically examined both sides of the sheet (presumably as part of my SAWING or CUTTING-A-BOARD-TO-FIT schema) and selected the good side to face into the house with the bad side to be exposed to the elements. After I had cut the panel and fitted it in place I noticed that several splinters had been torn out leaving ruts in the "good" side. I immediately saw the problem. Hand saws only cut in one direction. With hand saws, the downward motion does the cutting while the upward motion only repositions the cutting blade for another downward motion. I had cut the wood panel with the "good" side facing down. The downward cutting action has a tendency to tear splinters of wood out of the lower surface of the board. Since the good side was the lower surface, it suffered the loss of splinters. If I had to perform the same action again, I would not make the same mistake. I would cut the board with the good side facing up. However, what I learned was not just a simple special-

> ized patch to handle this particular instance of splintering. Since I knew the cause of the splintering, I knew that it would not always be a problem: it is only a problem when 1) the lumber is prone to splintering, 2) there is a "good" side of the board that is to be preserved, and 3) one is making a crosscut (across the wood's grain) rather than a rip cut (along the grain). Moreover, the solution is not always to position the wood with the good side up. My electric saber saw (also a reciprocating saw) cuts during the upward blade motion rather than the downward motion. Clearly, the solution when using the saber saw is the opposite: to position the board with the good side down. Now, these are not hard and fast rules: with a sufficiently poor quality sheet of plywood splintering would likely always be a problem. Rather, these are useful heuristics that lead to a refinement of the SAWING schema.

Note that this refinement to the SAWING schema is far more general than required to handle the particular problem that gave rise to it. The refinement contains contingencies relevant to the use of saber saws even though no saber saw was used in the immediate problem. This is possible because the refinement is driven by world model, not just the problem. The SAWING schema was altered by identifying and eliminating the offending cause in the underlying knowledge-based explanation of the phenomena.

2.2.4. Volitionalization

This situation involves transforming a schema for which there is no planner (like VEHICLE-ACCIDENT, ROULETTE, etc.) into a schema which can be used be a planner to attain a specific goal. Consider the following story:

> Herman was his grandfather's only living relative. When Herman's business was failing he decided to ask his grandfather for a loan. They had never been close but his grandfather was a rich man and Herman knew he could spare the money. When his grandfather refused, Herman decided he would do the old fellow in. He gave him a vintage bottle of wine spiked with arsenic. His grandfather died. Herman inherited several million dollars and lived happily ever after.

This story is a paraphrase of innumerable mystery stories and illustrates a schema familiar to all who-done-it readers. It might be called the HEIR-ELIMINATES-BENEFACTOR schema. It is produced via volitionalization by modifying the existing non-volitional schema INHERIT. INHERIT is non-volitional since there is no active agent. The schema simply dictates what happens to a persons possessions when he dies.

In this example, volitionalization parallels schema composition. One of the preconditions to INHERIT is that the individual be dead. The ELIMINATE-BENEFACTOR schema uses the schema MURDER to accomplish this. One major difference is that schema composition requires all volitional schemas. This parallelism need not always be present, however. Non-volitional to volitional transformation is also applicable to removing stochastic causal steps from a schema resulting in a volitional one.

2.3. Limits on Generalization

Basically, the generalization process is based on certain data dependency links established during understanding.

After a story is understood, the understood representation can be viewed as an explanation of why the events are plausible. For example, take the case of a kidnapping. KIDNAP is an instance of schema composition, not unlike RANSOM. Thus, the first kidnapping story seen by the system is understood as a THEFT followed by a BARGAIN. If the kidnapper is successful, the ransom is paid. For a system to understand this, it must justify that the person paying values the safety of the kidnapped victim

more that the ransom money. This justification is a data dependency (Doyle (1978)) link to some general world knowledge (e.g., that a parent loves his children). Now the event can be generalized so long as these data dependency links are preserved. Clearly, as long as the data dependencies are preserved, the underlying events will still form a believable whole.

Consider again the secondary effect elevation example of Fred trying to date Sue. The observed specific instance is John's interactions with Mary. Notice, however, that Fred did not simply copy John's actions. John actually made a date with Wilma while Fred only expressed an intention to date Lisa. This is not an earth-shaking difference, but in the context of dating it is extremely significant. In the normal DATE situation expressing an intention to date someone is not nearly so satisfying as an actual date. Once modified for the purpose of causing jealousy, however, expressing an intention for a date and actually carrying it out can be equally effective. That is, they both maintain the data dependency link for why we believe that Sue is in fact jealous.

Likewise, in the alteration example the schema for preserving one side of a board while sawing can be generalized. The resulting schema is applicable to circular saws, jig saws, etc. as well as hand saws. Again this is due to the preservation of a data dependency link: We believe that the wood's surface is preserved because the surface is supported by the rest of the board during deformation due to the saw's teeth. As long as we know which direction the teeth point on a saw, we know how to orient the board to preserve its good side.

2.4. Comparison to Previous Work

How does this method compare to other learning systems? There are a number of previous learning systems that spring to mind: Schank's MOPs, Selfridge's language learning model, Soloway's program to learn the rules of baseball and SRI's STRIPS system. The system outlined is strikingly different from Schank's and Selfridge's. It has some interesting similarities to Soloway's and one part of the STRIPS system.

While the domain of Schank's MOPs is similar to the described system, the learning technique used with MOPs is very different. The systems of Kolodner and Lebowitz both made "generalizations" but these are all of the correlational variety and might better be termed "specializations". IPP's generalization that Italian terrorists tend to shoot people in the knee caps, for example, is actually a correlational constraint noticed in the pre-existing terrorism MOP. The result is actually a specialized terrorism MOP to be applied only to Italian terrorist stories which makes a prediction about shooting in knee caps. Learning in both IPP and CYRUS is of this variety. Their approach precludes the kind of learning that extends a system's range of processing. Lebowitz's general terrorism MOP could not in principle be learned by his system. In the example outlined, the system learned an EXTORT schema without having a more general version already built in.

Selfridge's system was concerned with learning sentence structure and the names of already existing concepts. It learned, for example, that the words "put on" can refer to the already defined algorithmic concept "get

dressed in". The domain of my system is learning the original concepts. It might be interesting to explore how these ideas could be applied to language learning but that would not be the main thrust.

Soloway's system is similar to the one outlined here in that it has the flavor of one-trial or "insight" learning. Furthermore, he made use of general background goal information (in the form of notions such as competition) to aid in processing. However, the domain of learning baseball rules from game descriptions is very different from learning process schemata. Also, the purpose of his system is very different. It did not try to extend the range of its processing in an open-ended way. Rather, it tried to induce general rules from instances. In that sense it is more of an inductive inference system.

The MACROPS idea of SRI's are similar in that they result in new processing structures which can in turn be combined to form yet other structures. However, the domain of planning paths around blocks and through doors is much more constrained and simplified. Furthermore, the MACROPS structures were built from a successful planning search through the problem space, not in the midst of processing inputs. This makes STRIPS very inward motivated in its learning.

2.5. Conclusion

There are several concluding points

1) Explanatory schema acquisition does not depend on correlational evidence. Unlike some learning system (e.g., Winston (1970) and Fox and Reddy

(1977)) it is capable of one trial learning. It is somewhat similar to Soloway's view of learning (1977).

2) The approach is heavily knowledge-based. A great deal of background knowledge must be present for learning to take place. In this respect explanatory schema acquisition follows the current trend in AI learning and discovery systems perhaps traceable to Lenat (1976).

3) The learning mechanism is not "failure-driven" as is the MOPs approach (Schank (1980)). In that view learning takes place in response to incorrect predictions by the system. In explanatory acquisition learning can also be stimulated by positive inputs which encounter no particular problems or prediction failures.

4) The absolute representation power of the system is not enhanced by learning new schemas. This statement is only superficially surprising. Indeed, Fodor (1975) implies that this must be true of all self-consistent learning systems. Explanatory schema acquisition does, however, increase processing efficiency. Since all real-world systems are resource limited, this learning technique does, in fact, increase the system's processing power. Furthermore, it may indicate how Socratic method learning is possible and why the psychological phenomenon of functional fixedness is adaptive.

3. Understanding metaphor

3.1. Importance of metaphor

Metaphors are pervasive. It is nearly impossible to avoid metaphor in language use, even if the language is technical. For example, hydraulic metaphors are common in economics (e.g. economic pressure, cash flow, turning off the money supply, draining of assets, etc.). It is not possible to talk about love except through metaphor: love can be likened to a journey together, a meeting of minds, complementary shapes (as in fitting or belonging together), madness, falling into an abyss, transmitting and receiving on the same wavelength, and so on. Jackendoff (1975) has argued that metaphor is the basic process by which we acquire proficiency in abstract domains; he suggests that as infants, when we encounter a novel domain, we use existing sensory-motor schemas to form the basis of schemas suitable for understanding the abstract domain, and that this process can continue recursively, using existing abstract schemas as the basis for understanding novel abstract domains. Jackendoff therefore suggests that the surface similarity of "Mary kept the ring in a box" and "They kept the business in the family" reflects a deep similarity due to the derivation of the abstract domain of possession from the concrete domain of position.

Metaphors can be used to transfer complex combinations of information from one well-known domain to another less well known or completely unfamiliar one. Understanding metaphorical language first requires noting that the language is metaphorical, that is that it couldn't be literal descriptive text. This in turn requires an internal model of what is ordinary, expected, or possible, that a system can use to judge the plausibility of

novel language (see for example item d) in the introduction of this article.). Next, material from the "base domain", that is the domain in which the language has literal meaning, must be used to understand the "target domain", that is, the domain which is actually being described. This could be done in a number of ways, for example, by establishing links between the base domain of the metaphor and the target (novel) domain that the metaphor is being used to describe, or by copying base domain structures into a target domain. The result can become the basis for learning about a new domain (by transferring knowledge from the base domain selectively) or it may simply be that a metaphor allows one to express in a few words many notions about a target domain that would otherwise require a much lengthier exposition. Consider for example:

(S1) John ate up the compliments.

or

(S2) Robbie's metal legs ate up the space between him and Susie*.

Assuming that these sentences represented novel uses of the words "ate up", we might want a system to infer that in the first sentence John desired the compliments, eagerly "ingested" them with his mind, thereby making them internal and being given pleasure by them, and that in the second sentence, the distance between Robbie and Susie was being reduced to zero, just as an amount of food is reduced to zero when it is "eaten up".

In the following sections I will show methods which will make the correct interpretations of the two examples above. First, however, I must

---

*This is a slightly modified sentence from Isaac Asimov's I, Robot.

introduce "event shape diagrams", a new representation scheme for verb meaning, which is used centrally in this method for understanding novel metaphors**.

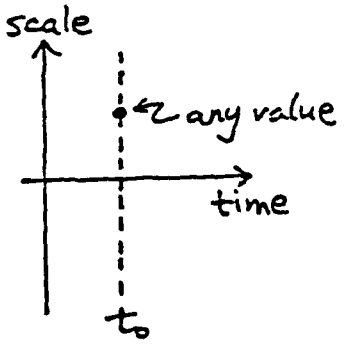
3.2. Event Shape Diagrams

In their simplest forms, event shape diagrams have a time line, a scale, and values on the scale at one or more points. Diagrams can be used to represent concurrent processes, causation, and other temporal relations by aligning two or more diagrams, as illustrated in Figure 1. Figure 1 shows the representation for "eat." Note that several simple diagrams are aligned, and that each has different kinds of scales, and different event shapes. The top scale corresponds to the CD primitive INGEST (Schank 1975). Causal relations hold between the events described in each simple diagram. The names for the causal relations are adopted from Rieger's CSA work (Rieger (1975)). The action INGEST stops in this default case where "desire to eat" goes to zero. "Desire to eat" sums up in one measure coercion, habit, and other factors as well as hunger. Typical values for amounts of food, time required to eat, and so on are also associated with the diagram, to be used as default values.

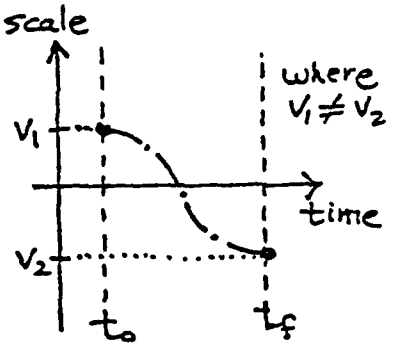
Many adverbial modifiers can be represented neatly: "eat quickly" shrinks the value of t_f-t_o with respect to typical values; "eat a lot" increases the values of q_o-q_f above typical values. Similarly "eat only

---

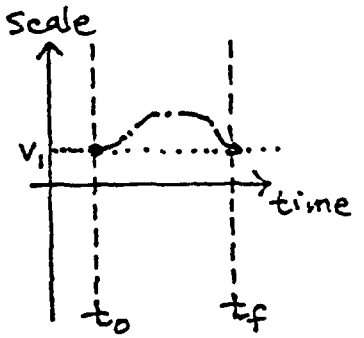
**Only verb-based metaphors will be treated here. These methods seem inappropriate for interpreting noun-based metaphors such as "John is a rat", or for "phenomenological metaphors", such as "I woke up in the morning with a sledge hammer banging in my head", as well as for others, no doubt. I have not attempted a taxonomy of metaphor types.



BE

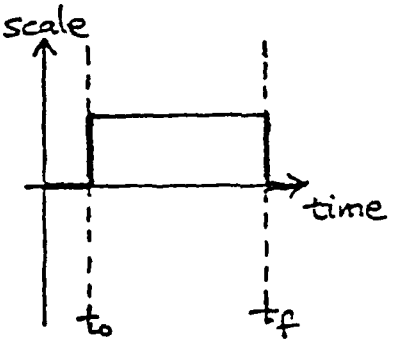


CHANGE



RECUR

Special case:
if scale$(t) = V_1$
for all t, $t_o \leq t \leq t_f$,
then the diagram represents
PERSIST



HAPPEN (if no agent)
DO (if agent)

Figure 1. Basic event shape diagrams.

half of one's meal," "eat very slowly," "eat one bite," etc. can be neatly represented. "Eat up" can be represented by making the

QUANTITY(food/IN1(food,digestive-tract(agent)))

go to zero before the DESIRE(agent,ACT1) goes to zero. This representation is shown in Figure 2.

The point of time from which events are viewed can also be clearly represented. Past tense (e.g. "we ate 3 hamburgers") puts "now" on the time line to the right of the action, while future tense puts "now" to the left of the action, and present progressive (e.g. "we are eating") puts "now" between t_o and t_f.

More levels of detail can be added if needed. For instance, the action diagram for eating ought to have links to more general event shape diagrams representing the typical daily eating habits of humans (three meals, one in the early morning, one around noon, and one in the early evening, plus between-meal snacks, coupled with diagrams representing the gradual onset of desire to eat after a meal); the diagram for "eating" should also should have links to more detailed event shape diagrams that expand upon the actions involved (eating involves many recurrences of putting food in one's mouth, biting, chewing, and swallowing, and the diagram for the amount of food inside the agent can reflect a series of stepwise changes as each mouthful is ingested.).

For more detail on event shape diagrams, see Waltz (1982).

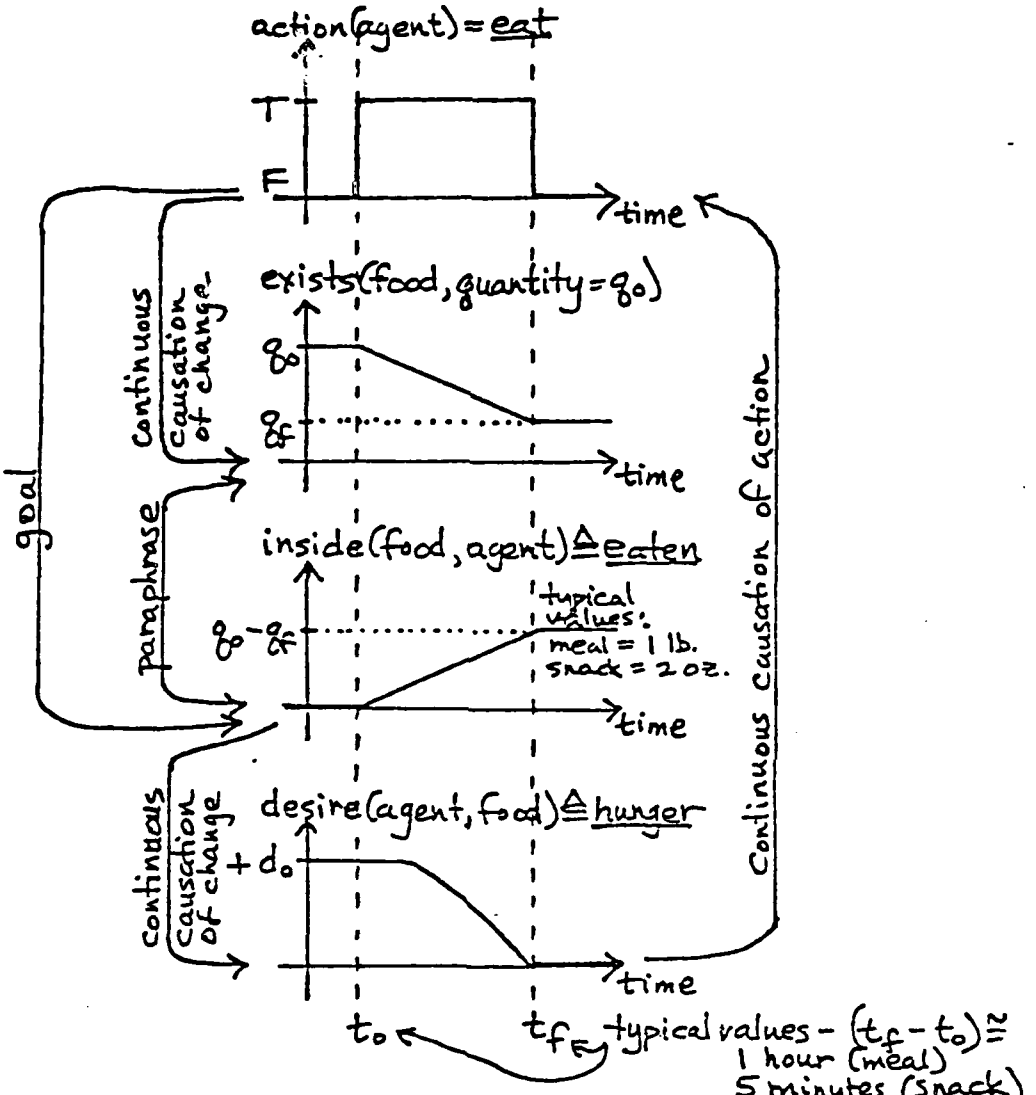


Figure 2. Event shape diagram for eat.

3.3. Metaphor with event shape diagrams

The interpretation of verb-based metaphors is based on the following general principles:

1) Both verbs and nouns have inherent selection restrictions. Thus, for the purposes of this example, "eat (up)" prefers that its semantic object be food, and foods of various kinds are marked by a preference to appear with certain actions, such as "eat", "buy", "grow", "prepare", "throw away", etc. (See Finin (1980) for discussion of "case frames" for nouns.)

2) Nouns are far less likely to be metaphorical than verbs. If a verb and object do not match each others' selection restrictions, the object should be taken as referring literally, and the verb as referring metaphorically. Thus, we can correctly predict that each of the following sentences is really about ordinary actions on food, even though literally these actions are very remote meanings for each of the verbs:

(S3) Mary destroyed the food. (= prepared badly or ate ravenously)

(S4) Sue made the food disappear. (= ate up rapidly)

(S5) John threw the food together. (= prepared rapidly)

3) Understanding of a verb-based metaphor involves a) selection of candidate meanings using the semantic object, b) matching the event shape diagrams of the candidate meanings with both the current context and the event shape diagrams of the actual verb in the sentence.

If there is more than one basic meaning candidate for a metaphorically-used verb (as in (S3) above) the most appropriate meaning is

selected by testing the various basic meanings in the current context to see which fits best. Once a basic meaning is selected, the event shape diagrams of this meaning are matched with the event shape diagrams of the actual verb used, and some meaning is transferred. The meaning transfer can take two forms: (1) modifying the basic meaning, in a manner similar to adverbial modification; and, (2) (more interestingly) superimposing certain portions of the event shape diagram for the verb actually used in the sentence onto the selected basic meaning.

This process should be clearer after I show examples of its operation on sentences (S1) and (S2).

3.4. An example

Consider the processing required to handle the metaphor in

(S1) John ate up the compliments.

Using principle (1) above, we first note that "ate up" prefers food of some kind as a semantic object, that "compliments" is not a food, and itself prefers an MTRANS-type verb (Schank 1975), in particular either "tell" or "hear". Next, using principle (2), we can judge that "compliments" refers literally, and so either "tell" or "hear" is probably the true basic verb. The event shape diagrams for "tell" and "hear" are shown in Figure 3. STM means "short term memory" and LTM means "long term memory". These terms are used here with their common sense (non-technical) meaning.

If the sentence appeared in context, we might be able to select the proper basic meaning by comparing the two possibilities with our current

expectations, but in this case, we have to rely on event shape diagram matching to determine the best choice.

Let us look first at trying to match "tell" with "eat up". In order to judge the quality of the match, we must first describe a scoring scheme. The scoring scheme used here is rather simple: it looks for scales that are the same, and matches them, provided the shapes of the scale are the same (i.e. both are changes in the positive direction, or both are occurrences, where an occurrence is defined as a change on some scale from a zero to a non-zero value, followed by a change back to zero again. In this case, MTRANS matches INGEST -- both are occurrences -- and

INTEND (agent,MTRANS(agent,compliment,STM(agent),STM(hearer)))

matches

INTEND (agent,INGEST(agent,food,[source],digestive-tract(agent)))

-- both are negative changes. There is a serious mismatch between these two, in that STM(hearer) does not match digestive-tract(agent) well, and these items are the goal portions of the DESIRE, the most important part.

Now consider the match between "hear" and "eat up" As before, MTRANS matches INGEST, but now the INTEND portion of "eat up" has no match. However, IN1 (compliment,STM(hearer)) matches IN2 (food,digestive-tract(agent)) very well -- both are the major scales of their respective verbs, and both have the same "shape", namely the occurrence shape, and finally, IN1 and IN2 are closely related binary predicates.

The understanding of the metaphor can now be addressed. Understanding

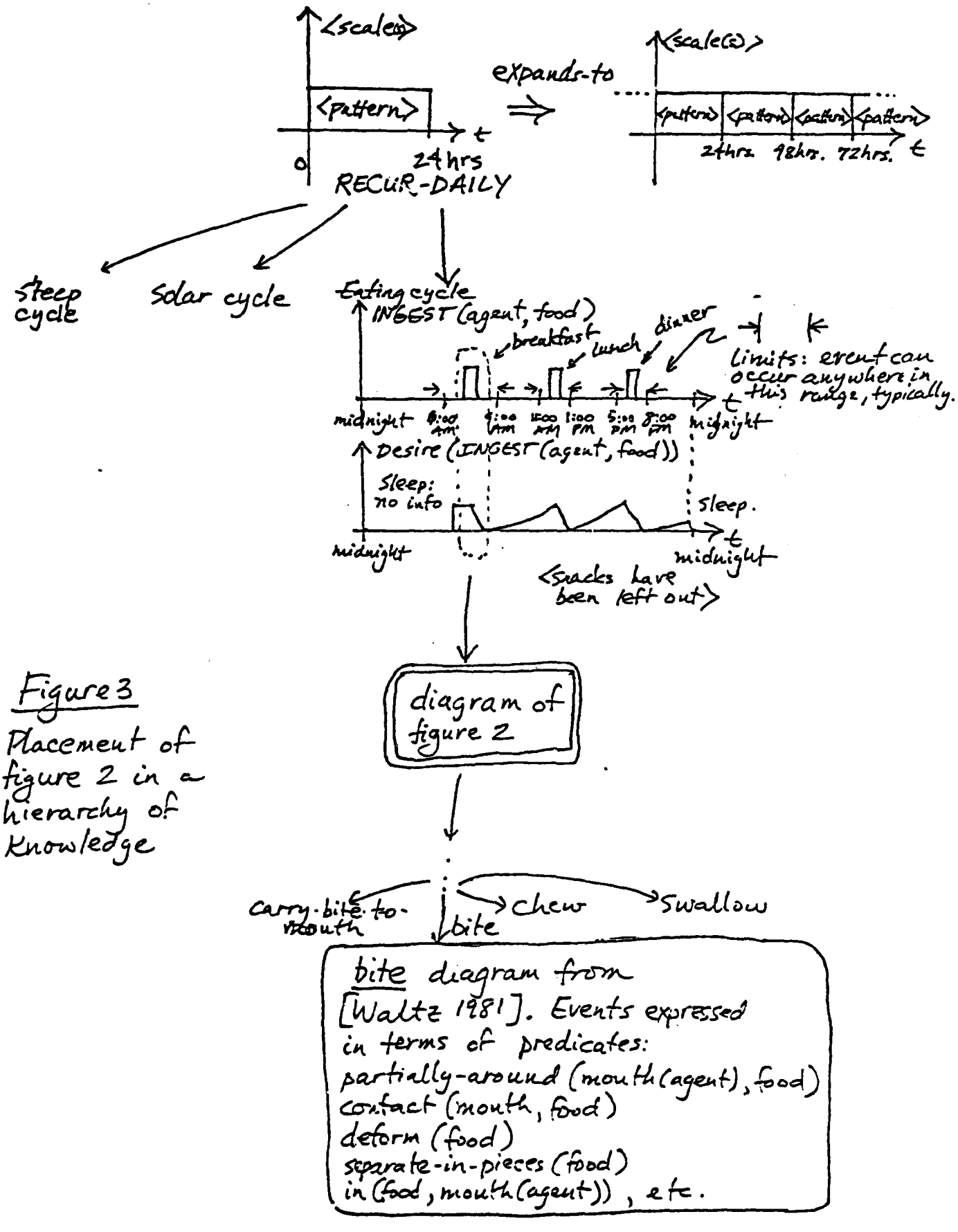


Figure 3

Placement of figure 2 in a hierarchy of knowledge

in this model is the transfer to hear of the "residue" of the meaning of eat up, where by "residue" I mean the portion of eat up that had no match with portions of hear. The residue in this case consists of the scales for DESIRE, INTEND, QUANTITY, and FEEL-PLEASURE that were associated with eat up. Theoretically, there are two main options for the mechanism that makes the transfer: (1) the scales may simply be added to the meaning of hear, or (2) some of these scales may already be present in latent or potential form as part of our understanding of hear, and the transfer would then consist of boosting their prominence, assigning a polarity to them, etc. Even within this single example, there are three kinds of issues that lead me to believe that option (2) is the right choice in general: first, it is difficult to understand why INTEND cannot be transferred to hear unless one realizes that hearing a particular item is not something we can ever intend in a causal sense; second, the transfer cannot be literal in any event -- for example we would not want to infer that compliments remain in our STM for a day, just because food may do so; and third, adverbial modification seems to already require scales to be present in latent form, as for example in

(S6) I heard the compliments with great pleasure.

Taking the second option, then, we can construct a meaning for (S1), as shown in Figure 4. Figure 4a shows the enriched version of hear used to receive the transferred material from eat up. Note that although the items below the dotted line are truly part of the meaning of hear, these items would not ordinarily be evoked when understanding the word hear, and that really, this version of hear represents three meanings, corresponding to

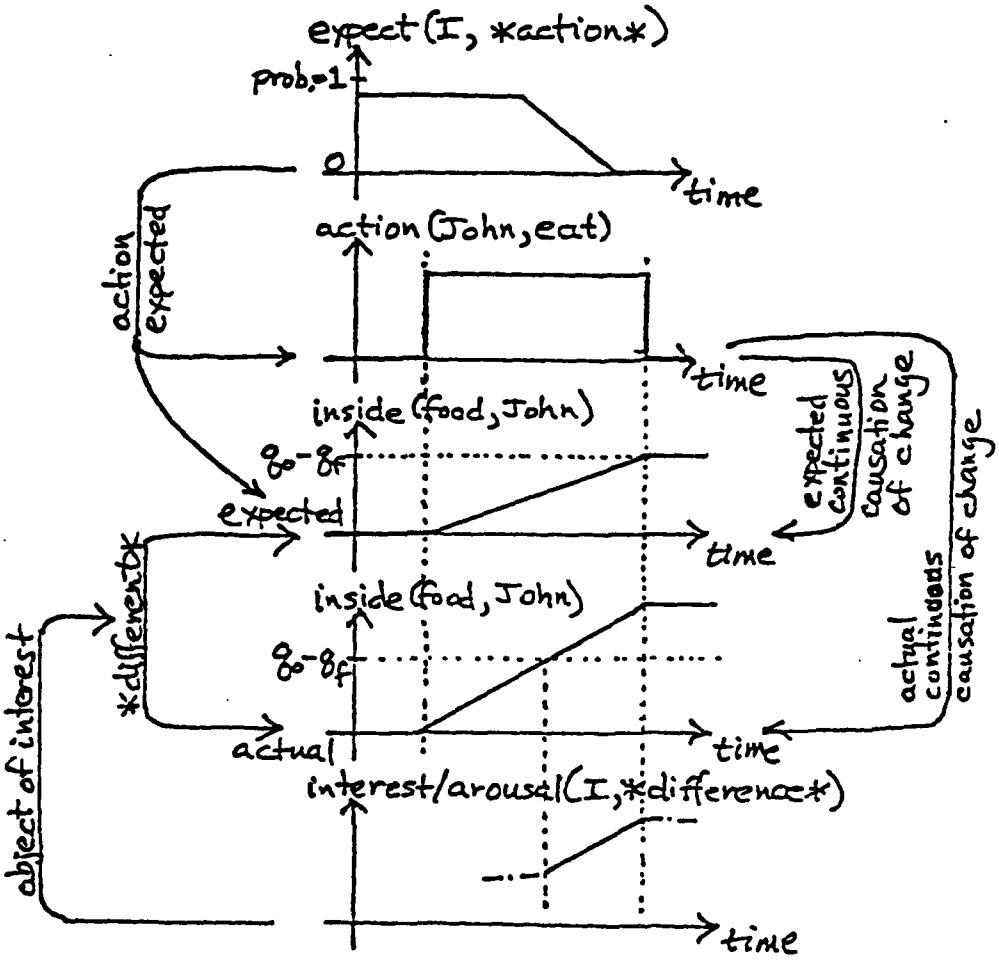


Figure 4. Representation of "I was surprised that John ate so much."

"hear", "hear with pleasure", and "hear with displeasure". It would clearly not be difficult to select "hear with pleasure" by matching with "eat up". Figure 4b shows the final meaning representation for (S1).

Example (S2)

(S2) Robbie's metal legs ate up the space between him and Susie.

can be understood using similar methods, though there are some interesting differences. The object of the verb in this case is "space" which is again not an appropriate object for use with "eat up". Again taking the semantic object as the item most likely to refer literally, space suggests that the true basic verb in the sentence ought to be PTRANS, that is, the physical transfer of an object through space. "Legs" also play an important part here, constraining the PTRANS to be either "run" or "walk" (this requires different processing methods that I have not yet investigated very thoroughly). For our purposes, "run" and "walk" look pretty much the same. There are some main variants that I believe ought to be represented differently, namely the meaning suggested by phrases such as run from (away from) x, run to (toward) y, run (without source or goal), run from x to y, and so on. These differ according to whether movement is stated with reference to a source, goal, neither or both, and whether or not the motion actually starts and/or ends at the source and goal points, or whether these specify only the direction of motion. In this case, the QUANTITY of food which goes to zero should make it possible to match the "run to" meaning.

So far, so good, but some interesting issues remain. First, there is little residue to transfer in this case, except for the intensification of the DESIRE to be at the goal. In fact, I don't think that this is bad, but

there are some inferences that I make in hearing (S2) that cannot be easily accounted for using this model. In particular, there is an analogy between taking bites and taking steps, and perhaps more important (and possibly related) (S2) seems to focus on the past progressive aspects of the action; to my mind the sentence is better paraphrased as "Robbie was running toward Susie" than as "Robbie ran to Susie". Overall, however, the account of the understanding of the two metaphors seems to capture roughly the right meanings in a natural and (to me) quite satisfying manner; the problems seem to require refinements to the method rather than complete rethinking.

3.5. Assessment

I do not want to claim that all metaphors can be handled by methods of the sort that have been described above. I do believe that the mechanisms suggested above are particularly good and natural for a reasonably rich class of metaphors. There still are holes in the theory, however. Consider the following sentence (due to Gentner (1980)):

(S7) The flower kissed the rock.

I have suggested that objects ought to be taken literally, and indeed, if we do so, we can obtain a reasonable reading, namely that a flower bent over and its "face" touched a rock gently. However, one could also take the verb literally, and take "rock" and "flower" metaphorically; In this case, the sentence could refer to a gentle woman literally kissing a tough man.

4. Conclusion

This work is just beginning. The examples we describe have been chosen to be types that commonly occur, so that rules needed to understand them can also be used to understand a much wider range of novel language. However, we must note that there is only so far that rules can take us: ultimately the power of systems will depend on the sheer amount of knowledge they have, knowledge which can be used as the base domain for new metaphors, and schemas that can be used to build yet more schemas. Therefore, to really achieve something resembling common sense, we will have to exercise our rules on whatever base of information we have, building a yet larger base on which the rules can operate recursively.

REFERENCES

Bobrow, D., R. Kaplan, M. Kay, D. Norman, H. Thompson and T. Winograd. 1977. "GUS, a frame driven dialog system," Artificial Intelligence, Vol. 8, No. 1.

Bobrow, D. and D. Norman. 1975. "Some principles of memory schemata," in Representation and Understanding, D. Bobrow and A. Collins (Eds.). Academic Press, New York.

Cercone, N. 1977. "A note on representing adjectives and adverbs," in Proc. IJCAI-77, MIT, Cambridge, MA, Aug. 1977, 139-40.

Chafe, W. 1975. "Some thoughts on schemata," in Proc. of the Workshop on Theoretical Issues in Natural Language Processing, Cambridge, MA, June.

Charniak, E. 1976. "A framed PAINTING: The representation of a common sense knowledge fragment," Cognitive Science, 4, pp. 355-394.

Charniak, E. 1977. "MS. MALAPROP, a language comprehension system," in Proc. 5th IJCAI, Cambridge, MA.

Charniak, E. 1978. "With spoon in hand this must be the eating frame," in Proc. 2nd Workshop on Theoretical Issues in Natural Language Processing, University of Illinois, Urbana, IL, July.

Colby, K.M., B. Faught and R. Parkinson. 1974. "Pattern matching rules of the recognition of natural language dialogue expressions," Stanford AI Lab, Memo AIM-234, June.

Cullingford, R. 1978. "Script application: Computer understanding of newspaper stories," Research Rept. 116, Yale Computer Science Department, New Haven, CT.

DeJong, G. 1979. "Skimming stories in real time: An experiment in integrated understanding," Research Rept. 158, Yale Computer Science Department, New Haven, CT.

DeJong,G. 1982. "Automatic Schema Acquisition in a Natural Language Environment," in Proc. 2nd Annual National Conference on Artificial Intelligence, Pittsburgh, PA, August 1982.

Doyle, J. 1978. "Truth maintenance systems for problem solving," MIT AI Tech. Rept. TR-419, MIT, Cambridge, MA.

Fahlman, S. 1979. NETL: A System for Representing and Using Real-World Knowledge. MIT Press, Cambridge, MA.

Finin, T.W. 1980. "The semantic interpretation of nominal compounds," Tech. Rept. T-96, Coordinated Science Lab, Univ. of Illinois, Urbana, March.

Fodor, J. 1975. The Language of Thought. Thomas Y. Crowell Company, New York.

Forbus, K.D. 1982. "Qualitative process theory," A.I. Memo No. 664, MIT AI Laboratory, Cambridge, MA, February.

Fox, M. and R. Reddy. 1977. "Knowledge-guided learning of structural descriptions," in Proc. 5th IJCAI, Cambridge, MA.

Gentner, D. 1980. Talk presented to the Conference of the Cognitive Science Society, Yale University, New Haven, CT, June.

Granger, R.H. 1977. "FOUL-UP: A program that figures out meanings of words from context," Proc. IJCAI-77, MIT, Cambridge, MA, August 1977, 172-8.

Jackendoff, R. 1975. "A system of semantic primitives," in Theoretical Issues in Natural Language Processing, R. Schank and B. Nash-Weber (Eds.). ACL, Arlington, VA.

Lebowitz, M. 1980. "Generalization and memory in an integrated understanding system," Computer Science Research Rept. 186, Ph.D. dissertation, Yale University, New Haven, CT.

Lenat, D.B. 1976. "AM: An artificial intelligence approach to discovery in mathematics as heuristic search," SAIL-AIM-286, Stanford University, July.

Minsky, M. 1974. "A framework for the representation of knowledge," MIT AI Rept. TR-306, MIT, Cambridge, MA.

Perrault, C.R. and P.R. Cohen. 1981. "It's for your own good: a note on inaccurate reference," in Elements of Discourse Understanding, Joshi, Sag and Webber (Eds.). Cambridge University Press, pp. 217-230.

Rieger, C. 1975. "The commonsense algorithm as a basis for computer models of human memory, inference, belief and contextual language comprehension," in Theoretical Issues in Natural Language Processing, R. Schank and B. Nash-Webber (Eds.). ACL, Arlington, VA, pp. 180-195.

Schank, R.C. 1975. "The primitive ACTs of conceptual dependency," in Theoretical Issues in Natural Language Processing, R. Schank and B. Nash-Webber (Eds.). ACL, Arlington, VA.

Schank, R. 1980. "Language and memory," Cognitive Science, 4, pp. 243-283.

Schank, R. and R. Abelson. 1977. Scripts Plans Goals and Understanding. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schmidt, C. and N. Sridharan. 1977. "Plan recognition using a hypothesize and revise paradigm: an example," in Proc. 5th International Joint Conference on Artificial Intelligence, pp. 480-486.

Soloway, E. 1978. "Learning = Interpretation + Generalization: A case study in knowledge-driven learning," COINS Technical Report 78-13, University of Massachusetts, Amherst, MA.

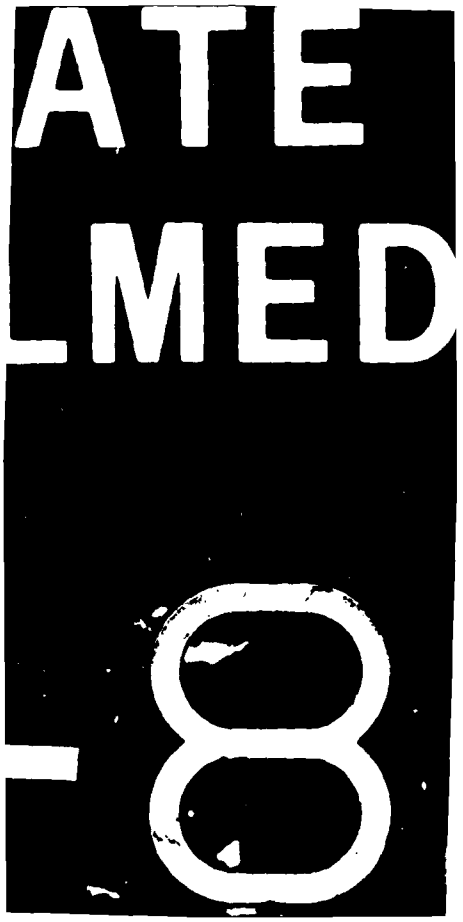
Waltz, D.L. 1981. "Toward a detailed model of processing for language describing the physical world," in Proc. IJCAI-81, Vancouver, B.C., Canada, August, pp. 1-6.

Waltz, D.L. 1982. "Event shape diagrams," Proc. 2nd Annual National Conference on Artificial Intelligence, Pittsburgh, PA, August.

Weizenbaum. 1966. "ELIZA - A computer program for the study of natural language communication between man and machine," Comm. ACM, Vol. 10, No. 8, pp. 474-480.

Wilensky, R. 1978. "Understanding goal-base stories," Ph.D. dissertation, Yale Computer Science Rept. 140, Yale University, New Haven, CT, September.

Winston, P.H. 1970. "Learning structural descriptions from examples," Rept. No. AI TR-231, MIT AI Lab, Cambridge, MA.


ATE
LMED
8